File No. 1410/7010-30
Form C28-0322-1

IBM Systems Reference Library

IBM 1410/7010 Operating System (1410-PR-155)
Basic Input/Output Control System—1410-IO-966

This publication provides 1410 and 7010 programmers with the information needed to write efficient programs incorporating the Basic Input/Output Control System.

The Basic Input/Output Control System, an integral part of the IBM 1410/7010 Operating System, can schedule, implement and control the transfer of data to and from core storage. It can also perform functions related to the transfer of data, such as error detection and correction.

Included in this publication are detailed descriptions of the DTF (Define The File) statements, the Basic Input/Output Control System macro-instructions, disk file processing using the Basic Input/Output Control System, and certain aspects of the Internal Operation of the IOCS that will enable the programmer to take advantage of the flexibility of IOCS.

MAJOR REVISION (November 1963)
This publication is a major revision of, and obsoletes, the publication, *IBM 1410/7010 Operating System; Basic Input/Output Control System,* Form C28-0322, and associated Technical Newsletters N28-1076 and N28-2011.

Copies of this and other IBM publications can be obtained through IBM Branch Offices.
Address comments concerning the contents of this publication to:
IBM Corporation, Programming Systems Publications, Dept. D91, PO Box 390, Poughkeepsie, N.Y.

Contents

Purpose of this Publication

This publication provides programmers with the information necessary to write efficient programs using the Basic Input/Output Control System of the IBM 1410/7010 Operating System.

Purpose of the IOCS

The Basic Input/Output Control System (hereafter referred to as the IOCS) provides programmers with a tested, efficient means of scheduling, implementing, and controlling the transfer of data between specified input/output devices and 1410 or 7010 core storage.

Specifically, the IOCS can perform the following functions:

1. Read data records or blocks of data records from the IBM 1402 Card Read Punch, IBM 1442 Serial Card Reader, IBM 1011 Paper Tape Reader, IBM 729 and 7330 Magnetic Tape Units, and IBM 1301 Disk Storage.
2. Write data records or blocks of data records on the 1402 Card Read Punch, IBM 1403 Printer, 729 and 7330 Magnetic Tape Units, and 1301 Disk Storage.
3. Detect error conditions, and correct those error conditions that lend themselves to correction.
4. Block and deblock data records.
5. Overlap read/write and processing operations.
6. Read, check, and write tape labels.
7. Provide exits to user-written routines such as end-of-file routines.
8. Detect requests for TELE-PROCESSING® interrupts and pass control to the Tele-Processing Supervisor so that such requests can be processed.
9. Provide for automatic rewinding, rewinding and unloading, or backspacing of the various reels of tape files at specific points in the user's program.

Prerequisites

It is assumed that readers of this publication have completed a basic course in programming the IBM 1410 or 7010 Data Processing Systems, and are familiar with the information contained in the following publications:

IBM 1410/7010 Operating System; Basic Concepts, Form C28-0318.

IBM 1410/7010 Operating System; The System Monitor, Form C28-0319.

IBM 1410/7010 Operating System; Autocoder, Form C28-0326.

Minimum Machine Requirements

The minimum machine requirements for programs incorporating the IOCS are contained in the publication, *IBM 1410/7010 Operating System; System Generation,* Form C28-0352.

Definition of Terms

Basic terms that appear frequently in this publication are defined here for the convenience of the reader.

Data Record: A collection of related facts, numbers, letters, or symbols that are treated as a unit and may be processed or produced by a computer.

Logical Record: A data record.

Card, Tape, Disk, and Printer Records: One or more data records brought together in such a way that they may be read or written by a single input/output instruction.

Physical Record: A card, tape, disk, or printer record.

Data File: A group of data records brought together for a specific purpose, such as grouping a company's accounts receivable.

Tape, Disk, Card Read, Card Punch, and Printer Files: Data files that are read from or written on, respectively, tape, disk, card read, card punch, or printer input/output devices.

Input and Output Files: Data files from which data records are read (input), or on which data records are written (output).

Basic Use of The IOCS

The purpose of this section is to provide the information necessary to write efficient programs that make use of the functions of the IOCS outlined under "Purpose of the IOCS" in the introduction to this publication. It should be noted, however, that the IOCS is capable of providing additional or expanded functions. These additional functions are discussed in a later section entitled "Extended Use of the IOCS."

IOCS Check List

Whether the programmer wishes to take advantage of the basic functions of the IOCS, or the basic functions plus selected additional functions, the following conditions must be met if the IOCS is to be used:

1. The configuration of the Resident IOCS must be defined at System Generation by means of the IOKDF macro-instruction and one or more DEVDF macro-instructions; or by means of the IOKDF macro-instruction, one or more DEVDF macro-instructions, and one or more DSKDF macro-instructions. (The IOKDF, DEVDF and DSKDF macro-instructions are covered in the *System Generation* publication.)
2. Each data file that is to be processed by the object program must be defined by means of a DTF statement included in the source program. (The DTF statement is discussed later in this publication.)
3. Every input/output area that is to be used by the object program must be defined by means of an Autocoder DA (Define Area) statement included in the source program. (Input/output areas are discussed later in this publication. DA statements are covered in detail in the *Autocoder* publication, and discussed, as they pertain to the IOCS, later in this publication.)
4. All user-written routines (e.g., end-of-file routines) that are to be referenced by the object program must be written and included in the source program. (The DTF statement entries that inform the IOCS of the existence of the various user-written routines, and the routines themselves, are discussed later in this publication.)
5. The specific functions the IOCS is to perform during execution of the object program must be requested by means of IOCS macro-instructions included in the source program. (The IOCS macro-instructions are discussed later in this publication.)

General Information

Information of a general nature, the understanding of which is considered a prerequisite to this discussion of a basic use of the IOCS, is presented here. The material covered includes: record forms acceptable to the IOCS, standard tape labels, input/output areas, DA statements, blocking and deblocking, work areas and indexing, end-of-file determination, and wrong-length-record checking.

Blocked Records

Data records should be grouped in blocks when they are frequently read or written and are not excessive in length, because blocked records can be read and written at a faster rate than unblocked records. This is because the number of times a tape drive must be started and stopped and the number of times disk operations must be executed, to read or write a given number of data records, is reduced when the records are blocked. Furthermore, the reduction in the number of tape interrecord gaps and disk record addresses permitted by blocked records increases the available storage capacity of a reel of tape or track of disk storage.

Blocking and Deblocking

The IOCS is capable of making data records that have been read into core storage in blocks available to the user's program as individual records (i.e., deblocking them).

The IOCS is also capable of grouping individual data records (i.e., blocking them) for transfer from core storage.

Record Forms Acceptable to the IOCS

The IOCS accepts four forms of data records, under the following conditions:

1. Records of the same file must have the same form, although the IOCS is capable of handling several files each of which uses a different record form.
2. Records moved from one area of core storage to another by the IOCS must terminate with a record mark, or the area from which the record is moved must terminate in a group mark with word mark.

The four forms of data records acceptable to the IOCS are Form 1, 2, 3, and 4 data records.

FORM 1 DATA RECORDS

Form 1 records are unblocked, fixed-length or variable-length records. Form 1 records should be used whenever:

1. Data records are exceptionally long.
2. Data records use a nonstandard format.
3. The programmer plans to block data records himself prior to transferring them from core storage.
4. The programmer does not wish to have wrong-length-record checks performed on unblocked, variable-length data records.

Examples of Form 1 records, as they appear on magnetic tape and in disk storage, are shown in Figures 1 and 2.

NOTE: The disk track formats shown in Figures 2, 4, 6, and 8 are read or written by the Read/Write Full Track With Home Address or the Read/Write Single Record instruction. These formats were chosen arbitrarily to simplify the figures. The programmer may format his disk tracks to be read or written by any 1301 read/write instruction.

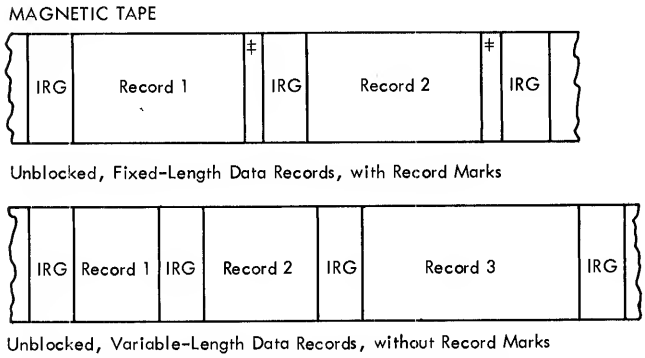


Figure 1. Form 1 Data Records on Magnetic Tape

FORM 2 DATA RECORDS

Form 2 records are blocked, fixed-length records. They should be used whenever data records are the same length. Examples of Form 2 records, as they appear on magnetic tape and in disk storage, are shown in Figures 3 and 4.

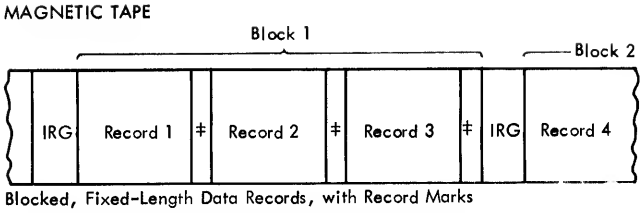


Figure 3. Form 2 Data Records on Magnetic Tape

FORM 3 DATA RECORDS

Form 3 records are identical to Form 1 records, with one exception. The first four positions of every Form 3 record contain a Block Character-Count. This count allows the IOCS to make wrong-length-record checks. When a Form 3 record is placed in an output area for transfer from core storage, the programmer must insert a group mark with word mark immediately to the right of the low-order position of the record, unless the low-order position coincides with the end of the output area. Examples of Form 3 records, as they appear on magnetic tape and in disk storage, are shown in Figures 5 and 6.

Block Character-Count: The Block Character-Count is a four-position field that must appear at the beginning of each Form 3 data record (see Figure 5) and at the beginning of each block of Form 4 data records (see Figure 7). This field contains an integer (right-justified) that specifies the number of positions of core storage required to contain the record or block of records. The positions occupied by the Block Character-Count and any terminal record marks that appear in the record or block of records are included in this number. Word marks may not appear over the three low-order positions of this field.

FORM 4 DATA RECORDS

Form 4 records are blocked, variable-length records that contain a Record Character-Count. In addition, the first four positions of each block of Form 4 records contain a Block Character-Count. This record form should be used when a file consists of a large number

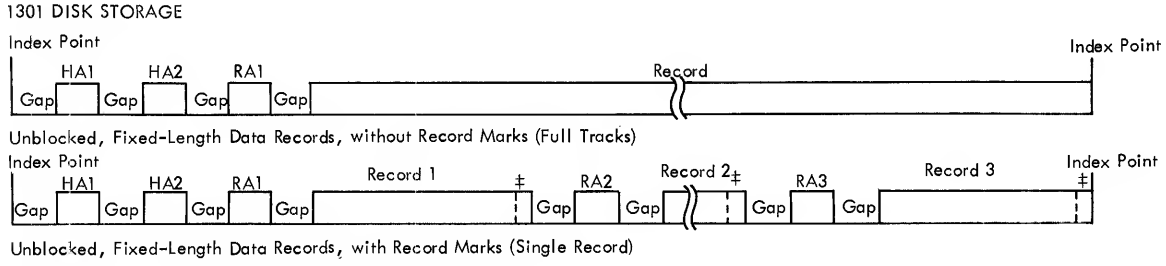


Figure 2. Form 1 Data Records in 1301 Disk Storage

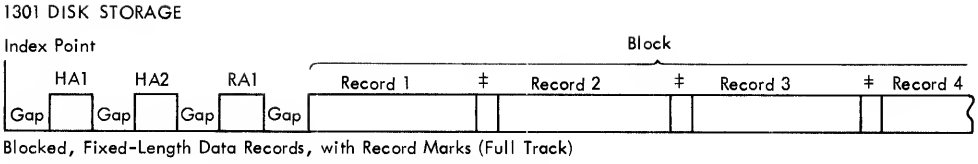


Figure 4. Form 2 Data Records in 1301 Disk Storage

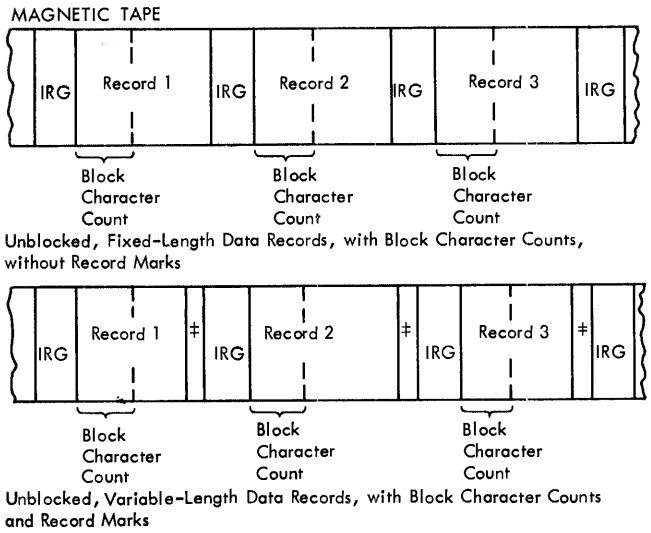


Figure 5. Form 3 Data Records on Magnetic Tape

of records that vary considerably in length. When a complete block of Form 4 records has been assembled in an output area for transfer from core storage, the IOCS places a group mark with word mark immediately to the right of the low-order position of the block.

Examples of Form 4 records, as they appear on magnetic tape and in disk storage, are shown in Figures 7 and 8.

Record Character-Count: The Record Character-Count is a field consisting of from one to five positions that must appear in every Form 4 data record handled by the IOCS (see Figure 7). The field contains an integer (right-justified) that specifies the number of positions of core storage required to contain the record. The positions occupied by the Record Character-Count and the record mark terminating the record (if any) are included in this number. This field must have a word mark over its high-order position, unless it is five positions in length. Word marks cannot appear over any other position. The low-order position of this field must appear at a fixed distance from the high-order position of each record in the file, and it cannot be signed minus.

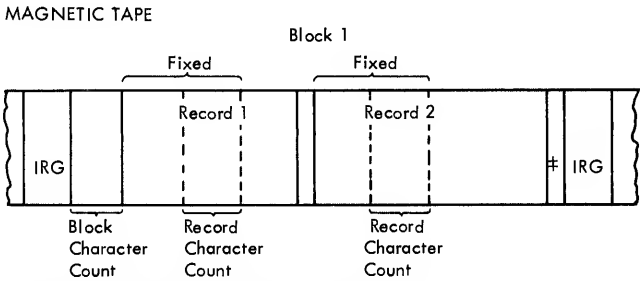


Figure 7. Form 4 Data Records on Magnetic Tape

RECORD FORM SUMMARY

The four forms of data records acceptable to the IOCS are summarized in Figure 9.

Standard Tape Labels

Optional tape label routines have been included in the IOCS to ensure that the correct tapes are made available to using programs, and to facilitate the maintenance of tape libraries. These tape label routines are based on use of 1410 80-character or IBM Standard 120-Character header and trailer labels. Header labels are the first record on each reel of tape, and

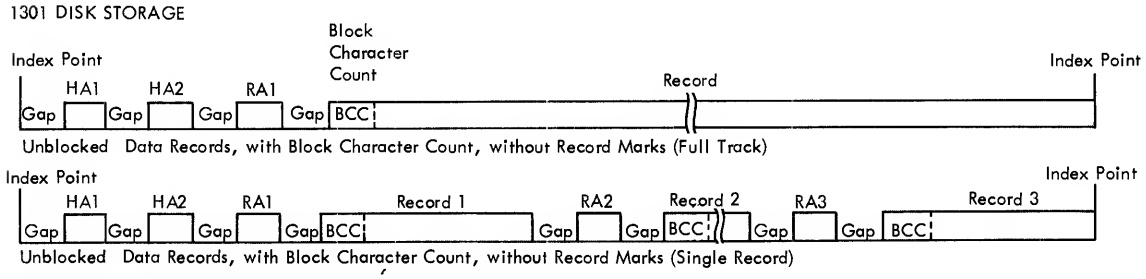
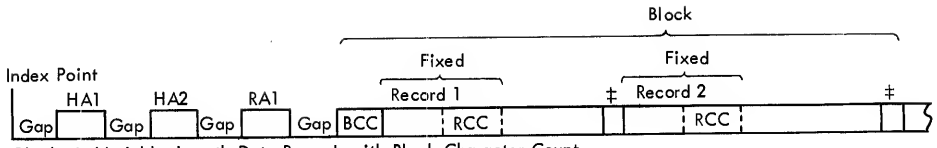


Figure 6. Form 3 Data Records in 1301 Disk Storage



Blocked, Variable-Length Data Records with Block Character Count, Record Character Counts, and Record Marks (Full Track)

Figure 8. Form 4 Data Records in 1301 Disk Storage

	Un-blocked	Blocked	Block		Record		Record Character Count	Block Character Count
			Fixed-Length	Variable-Length	Fixed-Length	Variable-Length		
Form 1	Must be				May be either			
Form 2		Must be	Must be		Must be			
Form 3	Must be				May be either			Must have
Form 4		Must be	May be either			Must be	Must have	Must have

Figure 9. Record Form Summary

serve to identify the tape. Trailer labels are, except for a tape mark, the last record on each reel of tape. They indicate whether or not a reel is the last reel of a file. The formats of the 1410 80-character and IBM Standard 120-Character header and trailer labels are discussed in the material that follows.

1410 80-CHARACTER TAPE LABELS

80-Character Temporary Header Labels: An 80-character temporary header label followed by a tape mark should appear on each reel of tape that is to become part of a file that uses 1410 80-character labels. The 80-character temporary header labels must conform to the format shown in Figure 10.

Field No.	Positions	Contents	Description
1	1-5	1HDRb	Label Identifier
2	6-10	xxxxx	Reel Serial Number
3	11-30	Blanks	
4	31-35	YYDDD	Creation Date
5	36-40	Blanks	
6	41-80	Miscellaneous	May be used to include other pertinent information

Figure 10. 1410 80-Character Temporary Header Label Format

Label Identifier. The label identifier (Field 1) consists of the characters 1HDRb. The label identifier indicates to the IOCS that the record containing these characters is a tape header label.

Reel Serial Number. The reel serial number (Field 2) is a five-digit number that is assigned to the reel when it enters the tape library of the using installation.

Field 3. This is a 20-position field that contains blanks.

Creation Date. The creation date (Field 4) is a five-digit number. The first two digits of this number specify the year; the remaining digits the day of the year this label was placed on the reel.

Field 5. This is a five-position field that contains blanks.

Miscellaneous. Field 6 can be used to include other information relevant to the label.

1410 80-Character Header Labels: The IOCS can replace 80-character temporary header labels with 1410 80-character header labels during execution of the user's object program. The 1410 80-character header labels must conform to the format shown in Figure 11.

Field No.	Positions	Contents	Description
1	1-5	1HDRb	Label Identifier
2	6-10	xxxxx	Reel Serial Number
3	11-15	xxxxx	File Serial Number
4	16-20	-xxxb	Reel Sequence Number
5	21-30	xxxxxxxxxx	File Identification
6	31-35	YYDDD	Creation Date
7	36-40	-xxxb	Retention Period
8	41-80	Miscellaneous	May be used to include other pertinent information

Figure 11. 1410 80-Character Header Label Format

Label Identifier. The label identifier (Field 1) consists of the characters 1HDRb. The label identifier indicates to the IOCS that the record containing these characters is a tape header label.

Reel Serial Number. The reel serial number (Field 2) is a five-digit number that is assigned to the reel when it enters the tape library of the using installation.

File Serial Number. The file serial number (Field 3) is a five-digit integer that is equal to the reel serial number of the first reel of the file.

Reel Sequence Number. The reel sequence number (Field 4) is a three-digit number preceded by a hyphen and followed by a blank character. It identifies the position within the tape file of this particular reel of the file (e.g., -007b appearing in this field would indicate that the reel is the seventh reel of the file).

File Indentification. The file identification (Field 5) consists of ten characters, and serves to identify the tape file to which the reel belongs.

Creation Date. The creation date (Field 6) is a five-digit number. The first two digits specify the year; the remaining digits the day of the year the label was placed on the reel.

Retention Period. The retention period (Field 7) is a three-digit number preceded by a hyphen and followed by a blank character. The three-digit number is the number of days the file is to be retained after the creation date appearing on the same reel. The IOCS can process retention periods of up to 365 days for this type of label. If a file is to be protected beyond this time, 999 should be placed in this field.

Miscellaneous. Field 8 may be used to include other information relevant to the label.

1410 80-Character Trailer Labels: The 1410 80-character trailer labels must conform to the format shown in Figure 12.

Field No.	Positions	Contents	Description
1	1-5	1EORb 1EOFb	Troiler-Label Identifier
2	6-10	xxxxx	Block Count
3	11-80	Miscellaneous	May be used to include other pertinent informotion

Figure 12. 1410 80-Character Trailer Label Format

Label Identifier. The label identifier (Field 1) consists of the characters 1EOFb if the reel is the last reel of the file, or 1EORb if the reel is not the last reel.

Block Count. The block count (Field 2) is the number of blocks of data records that appear on the reel of tape.

Miscellaneous. Field 3 may be used to include other information relevant to the label.

IBM STANDARD 120-CHARACTER TAPE LABELS

120-Character Temporary Header Labels: A 120-character temporary header label followed by a tape mark should appear on each reel of tape that is to become part of a data file that uses IBM Standard 120-Character labels. These temporary header labels must conform to the format shown in Figure 13.

Field No.	Positions	Contents	Description
1	1-5	1HDRb	Label Identifier
2	6-10	Blonks	
3	11-15	YYDDD	Creotion Date
4	16-30	Blonks	
5	31-35	xxxxx	Reel Sequence Number
6	36-100	Blonks	
7	101-120	Miscelloneous	May be used to include other pertinent informotion

Figure 13. IBM Standard 120-Character Temporary Header Label Format

Label Identifier. The label identifier (Field 1) consists of the characters 1HDRb. The label identifier indicates to the IOCS that the record containing these characters is a tape header label.

Field 2. This is a five-position field that contains blanks.

Creation Date. The creation date (Field 3) is a five-digit number. The first two digits specify the year; the remaining digits the day of the year this label was placed on the reel.

Field 4. This is a 15-position field that contains blanks.

Reel Serial Number. The reel serial number (Field 5) is a five-digit number that is assigned to the reel when it enters the tape library of the using installation.

Field 6. This is a 65-position field that contains blanks.

Miscellaneous. Field 7 can be used to include information relevant to the label.

IBM Standard 120-Character Header Labels: The IOCS can replace 120-character temporary header labels with IBM Standard 120-Character header labels during execution of the user's object program. The IBM Standard 120-Character header label must conform to the format shown in Figure 14.

IBM Standard 120-Character Trailer Labels: The IOCS can write IBM Standard 120-Character trailer labels at the end of each reel of tape. The format of these trailer labels is identical to that of the IBM Standard 120-Character header labels shown in Figure 14.

Label Identifier. The label identifier (Field 1) consists of the characters 1HDRb if the record is a header label; 1EORb if the record is an end-of-reel trailer label; and 1EOFb if the record is an end-of-file trailer label.

Field No.	Position	Contents	Description
1	1-5	1HDRb 1EORb 1EOFb	Label Identifier
	6	b	Blank
2	7-10	xxxx	Retention Period
3	11-15	xxxxx	Creation Date
4	16-25	xxxxxxxxxx	File Identification
5	26-30	xxxxx	File Serial Number
6	31-35	xxxxx	Reel Serial Number
	36	b	Blank
7	37-40	xxxx	Reel Sequence Number
	41	b	Blank
8	42-44	bbb	Reserved
9	45	x	Density Indicator
10	46	x	Checksum Indicator
11	47	x	Block Sequence Indicator
12	48	x	Tape Checking/Interpreting Technique Indicator
13	49	x	Tape Data Recording Technique Indicator
14	50	x	Tape Data Processing Technique Indicator
15	51-54	xxxx	Creating System
16	55	x	Record Format
17	56-60	xxxxx	Record Length
18	61-65	xxxxx	Block Size
19	66	x	Checkpoint Indicator
20	67-72	xxxxxx	Block Count
21	73-74	bb	Reserved
22	75-79	bbbbb	Reserved
23	80	b	Reserved
24	81-85	bbbbb	Reserved
25	86-91	bbbbbb	Reserved
26	92-100	bbbbbbbbb	Reserved
27	101-120	Miscellaneous	Any other pertinent information

Figure 14. IBM Standard 120-Character Header/Trailer Label Format

Retention Period. The retention period (Field 2) is a four-digit number (0001-9999) that represents the number of days the file is to be retained after the creation date specified in Field 3.

Creation Date. The creation date (Field 3) is a five-digit number. The first two digits of this number specify the year, the remaining digits the day of the year the file, of which this reel is a part, was created.

File Identification. The file identification (Field 4) consists of ten characters that identify the file to which the reel of tape belongs.

File Serial Number. The file serial number (Field 5) is a five-digit integer equal to the reel serial number of the first reel of the file.

Reel Serial Number. The reel serial number (Field 6) consists of five alphameric characters, other than blank characters, assigned to the reel when it enters the tape library of the using installation.

Reel Sequence Number. The reel sequence number (Field 7) is a four-digit number that identifies the position within the tape file of this particular reel (e.g., 0007 appearing in this field would indicate that this reel is the seventh reel of the file).

Reserved. Field 8 is left blank. It is reserved for future IBM programming use.

Density Indicator (Field 9), Check Sum Indicator (Field 10), Block Sequence Indicator (Field 11), Tape Checking/Interpreting Indicator (Field 12), Tape Data Recording Technique Indicator (Field 13), and Tape Data Processing Technique Indicator (Field 14). Fields 9-14 are not applicable. The IOCS enters a zero in each.

Creating System. Field 15 is used to specify the Data Processing System on which the file was created. The IOCS enters 1410 or 7010 (whichever is applicable) in this field.

Record Format. Field 16 is used to indicate the data record form used by the file. Form 1, 2, 3, or 4 records are represented in this field by B, F, W, or X, respectively.

Record Length. Field 17 is used to specify data record length. For fixed-length data records, the IOCS enters the number of characters in each record. For variable-length records, the IOCS enters the number of characters in the largest possible record.

Block Size. Field 18 is used to specify block size. For blocked, fixed-length records, the IOCS enters the number of data records in each block. For blocked, variable-length records, the IOCS enters the maximum number of characters that can be contained in a block. If the file consists of unblocked records, this field contains zeros.

Checkpoint Indicator. This indicator (Field 19) is not applicable. The IOCS enters zero in this field.

Block Count. Field 20 is used to specify the block count (i.e., the number of blocks of records on the reel). The block count does not include header labels, trailer, or tape marks. The block count is of significance only in a trailer label. In header labels this field is left blank.

Reserved. Fields 21 through 26 are reserved for future IBM programming use. The IOCS leaves these fields blank.

Miscellaneous. Field 27 may include any information relevant to the label.

Standard Tape Formats

When a tape file uses standard labels, the IOCS can accept only the tape formats shown in Figure 15.

Format A	Format B
120 Character Header Label Tape Mark Data Tape Mark 120 Character Trailer Label Tape Mark	80 Character Header Label Data Tape Mark 80 Character Trailer Label Tape Mark

Figure 15. Tape Formats Acceptable to the IOCS (Standard Labels)

Input/Output Areas

To read data records from or write data records on a data file, at least one input/output area must be provided in core storage to contain the data records that have been read or that are to be written.

The IOCS permits the programmer to specify more than one input/output area for each file so that he may take advantage of the Processing Overlap special feature.

The number of input/output areas used for each file can have a direct bearing on the efficiency of the program. When selecting the number of areas that are to be used, the programmer should consider the following:

1. Whether the program is input/output limited. If the program has to wait for input/output operations to provide records for processing or to write records that have already been processed, the use of multiple input/output areas can help reduce the running time of the program.
2. The number of files in the program. If there are many files, the number of input/output areas specified per file may have to be limited to conserve core storage.
3. The activity of the file. When the core storage available for input/output areas is limited, the major portion of that core storage should be allocated to areas that are to be used by the most active files.

Single and multiple input/output areas are discussed separately, in the material that follows, to aid the user in selecting the most suitable number of areas for each file.

ONE AREA

When one input/output area is specified for a file, the program must ordinarily wait until an input/output operation has been executed before processing the next record. For this reason, the programmer should normally avoid specifying a single area unless one or more of the following conditions apply:

1. Core storage is limited by other program requirements.
2. The file does not contain many records.
3. The file is seldom referenced by the program.
4. The file is read from or written on a unit-record input/output device.
5. The IOCS does not allow the use of more than one area. (Any such restriction is noted in the appropriate section of this manual.)

MULTIPLE AREAS

The pause to execute an input/output operation noted above can largely be eliminated by using multiple areas. When processing is completed in one area, the program begins processing the data record or records in the next area. At the same time, the IOCS reads into, or writes out of the area in which processing was just completed. Under these conditions processing is virtually continuous.

DA (Define Area) Statements

Autocoder DA statements are used to reserve and define all input/output areas used by programs incorporating the IOCS. DA statements used to define input/output areas associated with unit-record or tape files (Figure 16) must have the letter G preceded by a comma (, G) as the last entry in the operand field of the header line. This causes a group mark with word mark to be placed immediately to the right of the low-order position of the area. An example of the area this statement will reserve in core storage is shown in Figure 17.

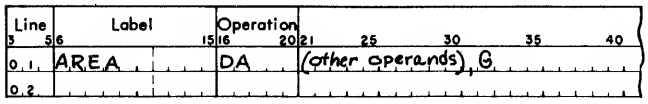


Figure 16. Unit-Record/Tape DA Statement

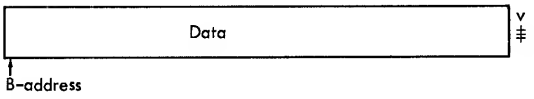


Figure 17. Input/Output Areas (Unit-Record/Tape Files)

Work Areas and Indexing

If a file consists of blocked records or uses multiple input/output areas, the programmer must decide whether it is more economical to process each data record in the input/output area(s) associated with the file by indexing the area(s), or to move each record into a work area and process it there. A general rule is available to guide the programmer. The number of positions of core storage occupied by an input/output area associated with the relevant file should be divided by four. If the result is less than the number of indexed references required to process the data in that area, a work area should be used. If the result is greater, indexing should be used.

End of File

The IOCS determines that the end of a tape file has been reached when any of the conditions noted below has occurred.

FILE TYPE	CONDITION
Tape Input	The file makes use of standard trailer labels, and a 1EOFb trailer label identifier has been encountered on a reel of the file.
Tape Input	The file does not use labels, and a tape mark has been encountered on a reel of the file.

Wrong-Length Records

The IOCS is capable of detecting wrong-length-record conditions. It performs this function, depending on the type of file, as follows:

1. For disk files, the IOCS checks the length of the record, as specified on the appropriate disk format track, against the size of the input/output area into which the record is read or from which it is written.
2. For input files that consist of Form 1 or Form 2 data records, a check is made to determine whether the interrecord gap following the record or block of records coincides with the group mark with word mark that terminates the input area into which the record or block was read. If the interrecord gap occurs before or after the group mark with word mark, a wrong-length-record condition is indicated.
3. For tape files that consist of Form 3 or Form 4 data records, the IOCS references the Block Character-Count that precedes the relevant record or block of records to perform the wrong-length-record check. (The IOCS cannot make wrong-length-record checks for tape files consisting of unblocked, variable-length records that do not include a Block Character-Count.)
4. For unit-record files, the IOCS determines if the length of the relevant input/output area coincides with the size of the appropriate buffer in the IBM 1414 Input/Output Synchronizer.

DTF Statement

The primary function of the DTF statement is to define the characteristics of the data file for which it was written. Each DTF statement consists of five or more entries. These entries may be broken down into those required for a basic use of the IOCS and those required for an extended use of the IOCS. The basic DTF entries are discussed in this section; the extended DTF entries are discussed in a later section.

GENERAL FORMAT

The DTF statement is written in the following manner. The code DTF is entered in the operation field of one line of the Autocoder coding sheet, and the name of the file that is to be defined is entered in the operand field of the same line. The entries relevant to this file are then listed on succeeding lines of the coding sheet. These entries must be selected from the entries discussed in this section or in the section on the extended use of the IOCS. As each entry is included, the operation field of the affected line of the coding form must be left blank. Operands of the same entry must be separated by commas, unless otherwise noted in the discussion of that entry. Labels included as operands of an entry may be address modified.

All entries may be followed by comments. These comments must be separated from the entries by a minimum of two consecutive blanks.

There are no restrictions on where the DTF statement defining a particular file must appear in the user's source program.

A DTF statement cannot be written for a file that refers to an input/output unit that has been designated the system's Standard Input Unit (SIU), Alternate Input Unit (AIU), Standard Punch Unit (SPU), Standard Print Unit (SPR), or System Operating File (SOF). (See the *System Monitor* publication for a discussion of these units.)

Basic DTF Entries

The DTF entries required to make a basic use of the IOCS are as follows:

DTF Header Line	RECSIZE
SYMUNIT	INDEX
FILEFORM	PADCHAR
BLOCKSIZE	ERRCHECK
IOAREAS	ERROPTNS
EOFADDR	RWDOPTNS
MODE	LABEL
ORDER	CHECK

DTF HEADER LINE

This entry is required. It must appear as the first entry of every DTF statement. It consists of the code DTF entered in the operation field of the coding sheet and the name of the data file that is to be defined entered in the operand field (see Figure 18).

The name of the file cannot be a linkage symbol. (Linkage symbols are discussed in the *System Monitor* publication.)

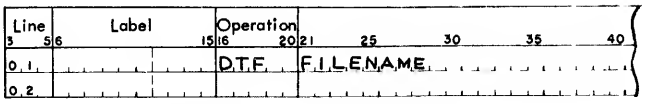


Figure 18. DTF Header Line

SYMUNIT

This entry is required. The first operand of this entry indicates the type of input/output device used by the file. The first operand must be one of the following:

OPERAND	EXPLANATION
1301	The file is read from or written in 1301 Disk Storage.
TAPE	The file is read from or written on magnetic tape units.*
READER	The file is read from a 1402 Card Read Punch or a 1442 Serial Card Reader.
PUNCH	The file is punched on a 1402 Card Read Punch.
PRINTER	The file is printed on a 1403 Printer.
1011	The file is read from a 1011 Paper Tape Reader.

The second operand of the SYMUNIT entry is the name of the symbolic unit that is to be used for the file. This name is used to assign a physical unit to the file. (See the *System Monitor* publication for discussions of symbolic units and physical units.) An example of how this entry is coded is shown in Figure 19.

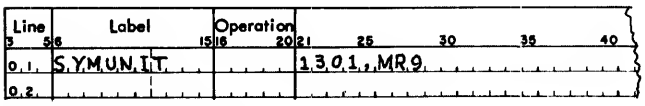


Figure 19. SYMUNIT Entry

FILEFORM

This entry is required. The operand of this entry indicates the data record form used by the file. The operand must be 1, 2, 3, or 4 to indicate, respectively, Form 1, 2, 3, or 4 data records. An example of how this entry is coded is shown in Figure 20.

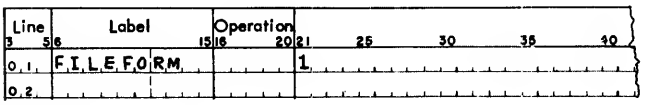


Figure 20. FILEFORM Entry

BLOCKSIZE

This entry is required. The operand of the BLOCKSIZE entry is an integer equal to the number of positions of core storage occupied by any one input/output area associated with the file. The group mark with word mark that terminates the area, the eight-digit disk track address, and eight-digit disk record address (if any) that precede the area are not included in this integer. Figure 21 shows an example of how this entry is written.

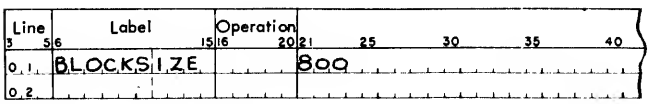


Figure 21. BLOCKSIZE Entry

IOAREAS

This entry is required for a basic use of the IOCS. It is used to indicate to the IOCS which input/output areas are to be associated with the file.

The IOAREAS entry may have a minimum of one, and a maximum of three operands. Each operand is the label of an input/output area that is to be associated with the file. Figure 22 shows an example of how this entry is written. Every input/output area whose label appears as an operand of this entry must be defined by an Autocoder DA statement. (If the programmer wishes to use more than three input/output areas to process a file, he must utilize the DTF FILELIST entry. This entry is discussed under "Extended Use of the IOCS.")

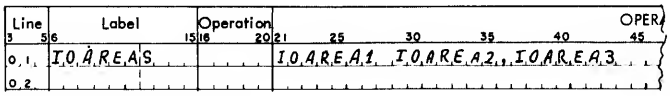


Figure 22. IOAREAS Entry

EOFADDR

This entry is required for all tape and unit-record input files. The operand of the EOFADDR entry is the label of the end-of-file routine the programmer has provided for the file. (This routine is entered whenever the IOCS determines that an end-of-file condition has occurred on the file.) An example of how this entry is coded is shown in Figure 23.

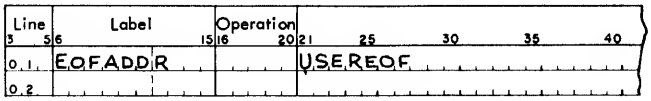


Figure 23. EOFADDR Entry

MODE

This entry is required for files that are read or written in Load mode. It is also required for tape files that are read or written in odd parity. Two of the possible operands of the MODE entry are LOAD and ODD. The LOAD operand indicates that the file is to be read or written in Load mode. The ODD operand indicates the file is to be read or written in odd parity. These operands may be entered in any order, and either operand may be omitted if it is not appropriate. See Figure 24 for an example of how this entry is written.

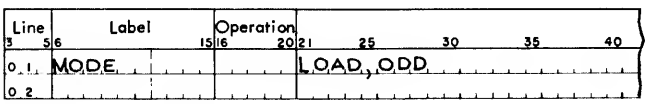


Figure 24. MODE Entry

ORDER

This entry is required for all files except tape files, unless otherwise noted in the discussion of operands that follows. The operand of the ORDER entry supplies the IOCS with the low-order character of the x-control field of the input/output instruction required to process the file. It must be one of the following:

OPERAND	EXPLANATION
0, 1, or 2	The file is an input card file and 0, 1, or 2 is the stacker into which cards read from the file are to be selected. If 1 is the appropriate operand and the entry is optional.
0, 4, or 8	The file is an output card file and 0, 4, or 8 is the stacker into which the punched cards that belong to the file are to be selected. If 4 is the appropriate operand, the entry is optional.
0	The file is a 1403 Printer file and the Write A Line instruction is to be used to process the file. If this operand is appropriate, the entry is optional.
1	The file is a 1403 Printer file and the Write Word Marks instruction is to be used to process the file. (The Write A Line and Write Word Marks instructions are explained in the publication, *IBM 1410 Principles of Operations,* Form A22-0526.)
1	The file is a 1301 disk file that is to be read/written by means of the Read/Write Single Record instruction.
2	The file is a 1301 disk file that is to be read/written by means of the Read/Write Full Track Without Record Addresses instruction. If this operand is appropriate, the entry is optional.
5	The file is a 1301 disk file that is to be read/written by means of the Read/Write Full Track With Home Address instruction.
6	The file is a 1301 disk file that is to be read/written by means of the Read/Write Full Track With Addresses instruction.
@	The file is a 1301 disk file that is to be read/written by means of the Read/Write Cylinder instruction.

The 1301 Disk Storage input/output instructions are discussed in detail in the publication, *IBM 1301 Disk Storage with IBM 1410 and 7010 Systems,* Form A22-6670.

An example of how the ORDER entry is coded, is shown in Figure 25.

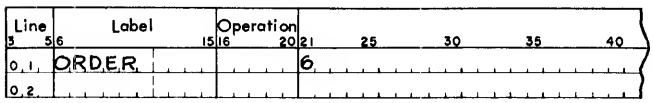


Figure 25. ORDER Entry.

RECSIZE

This entry is only required for files that consist of Form 2 or Form 4 records. If the file consists of Form 2 records, the operand of the RECSIZE entry specifies the number of characters that appear in each data record of the file. If the file consists of Form 4 records, the operand of the RECSIZE entry specifies the position, in each data record, of the low-order digit of the Record Character-Count. (The first position of a Form 4 tape record is considered 1, not 0, when determining the position of the low-order digit of the Record Character-Count.) An example of how this entry is coded is shown in Figure 26.

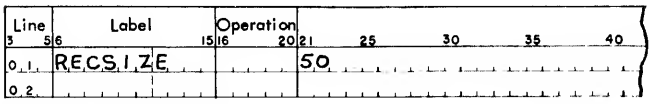


Figure 26. RECSIZE Entry

INDEX

This entry is required for a basic use of the IOCS, if: (a) the file uses blocked records or multiple input/output areas, and (b) the programmer plans to access data records in the input/output areas associated with the file.

The operand of the INDEX entry is X1-X12, depending on the index register (1-12) the programmer wishes to utilize. If this entry is included, and the file is an input file, the IOCS places, in the index register specified, the high-order address of the data record currently available to the using program. (See Figure 27.)

If this entry is included and the file is an output file, the IOCS places, in the index register specified, the high-order address of that portion of the current output area that is available to the using program. (See Figure 28.)

An example of how this entry is coded is shown in Figure 29.

PADCHAR

This entry is not required. It is relevant only if the file consists of blocked, fixed-length records (i.e., Form 2 data records). The PADCHAR entry (Figure 30) causes the IOCS to pad the last block of the file with the character entered as the operand of this entry. The character entered may be any character except ≠, ≢, *, ¢, or ¬, unless the file is to be sorted by the Generalized Tape Sort which is an integral part of the 1410/7010 Operating System. In this case, the character entered as the operand of this entry must be the character 9. If this entry is omitted, the IOCS pads with blank characters. (Blank characters are also acceptable to the Generalized Tape Sort.)

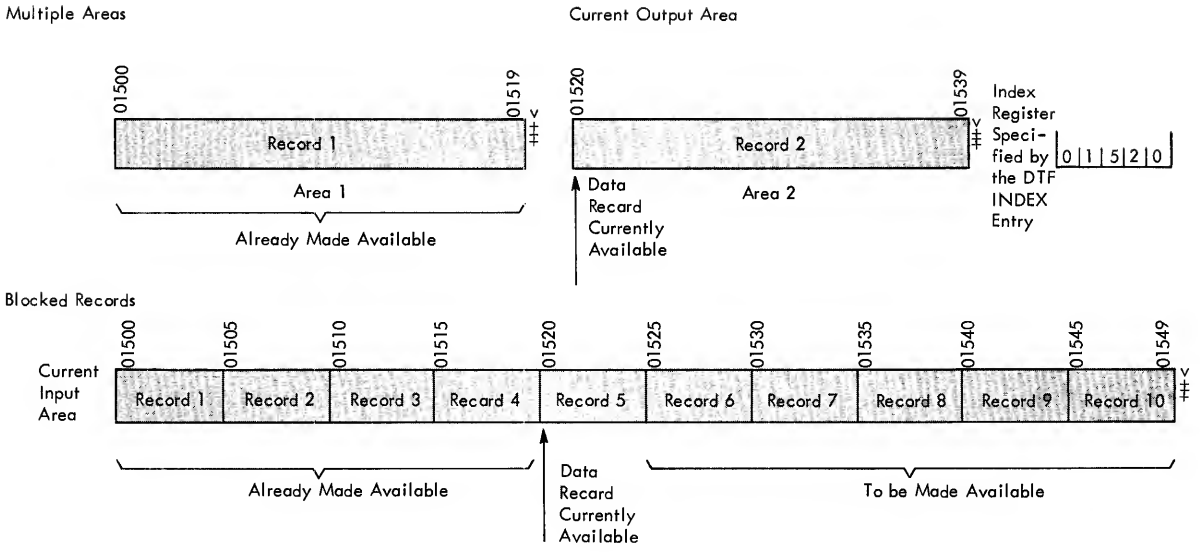


Figure 27. DTF INDEX (Input Files)

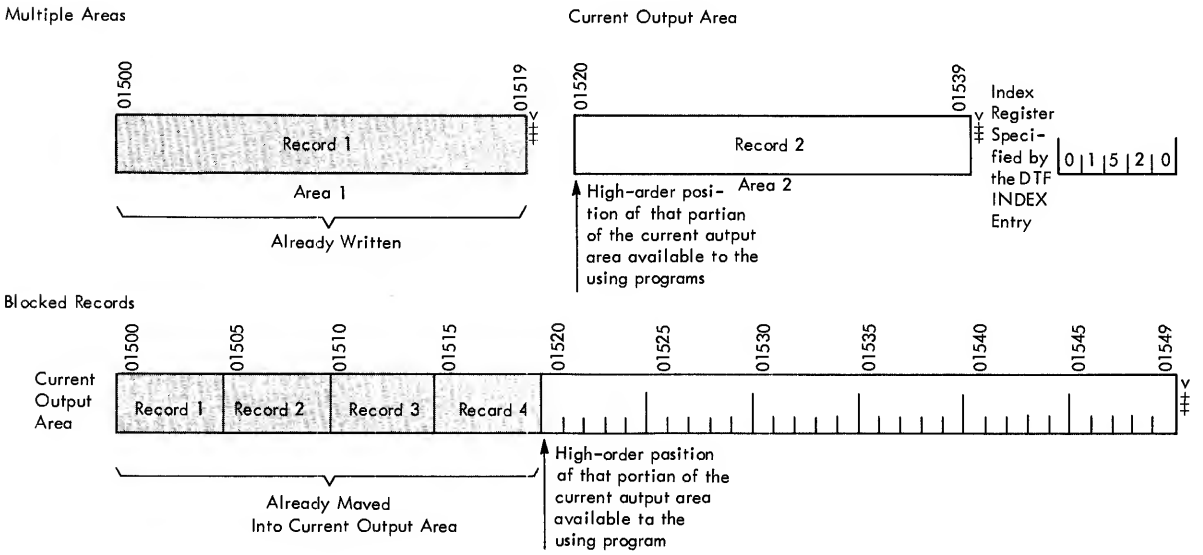


Figure 28. DTF INDEX (Output Files)

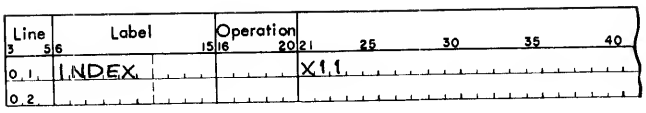


Figure 29. INDEX Entry

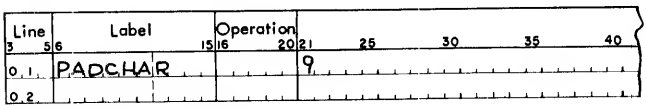


Figure 30. PADCHAR Entry

ERRCHECK

This entry is not required. It may be used to cause the IOCS to check for wrong-length records and/or to execute Write Disk Checks. The operands of the ERRCHECK entry, and the operations those operands cause the IOCS to perform, are as follows:

OPERAND	OPERATION
WLR	This operand causes the IOCS to check for wrong-length records. (It is recommended that this operand be specified for all input files, all unit-record output files, and all tape output files that consist of Form 3 or Form 4 records.) If this operand is omitted, the IOCS does not interrogate the Wrong-Length-Record channel status indicators.
WDC	This operand causes the IOCS to execute a Write Disk Check after every Write Disk operation performed on the file.

One or both of these operands can be entered, and in any order. An example of how this entry is coded is shown in Figure 31.

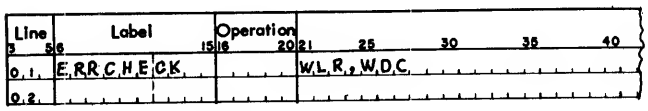


Figure 31. ERRCHECK Entry

ERROPTNS

This entry is not required. It should appear only if the file is an input file. It allows the programmer: (a) to accept all records read by input operations executed on the file that resulted in error conditions which the IOCS error procedures could not correct, or (b) to skip such records. The operands of this entry are:

OPERAND	FUNCTION
ACCEPT	This operand causes the IOCS to handle all uncorrectable, erroneous records that occur on the file as if they were error free (i.e., release them to the using program as the IOCS would a record read into core storage without error).
SKIP	This operand causes the IOCS to read the next logical record from the file into the input area that contains the uncorrectable, erroneous record, thereby destroying that record.

If this entry is omitted, the IOCS processes erroneous records as if they were error free.

Each of the operands discussed excludes the other. For this reason, only one may be specified in the same ERROPTNS entry. Figure 32 shows an example of how this entry is coded.

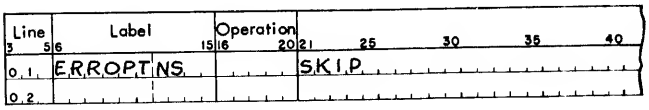


Figure 32. ERROPTNS Entry

RWDOPTNS

This entry is not required. It is of significance only if the file is a tape file. This entry allows the programmer to specify which reels of the file (if any) are to be rewound, rewound and unloaded, or backspaced — and at what point in the using program this action is to be taken.

The *operands* of this entry indicate what action is to be taken. The *operand fields* in which these operands appear indicate which reels of the file are to be acted upon and the time at which this action (if any) is to be taken. The four *operands* that can appear in the operand fields of this entry are:

OPERAND	ACTION
R	This operand causes the IOCS to rewind the tape reels indicated, at the time indicated.
U	This operand causes the IOCS to rewind and unload the tape reels indicated, at the time indicated.
B	This operand causes the IOCS to backspace the tape reels indicated, at the time indicated.
N	This operand causes the IOCS to take no action on the tape reels indicated, at the time indicated.

The four *operand fields* of this entry are:

OPERAND FIELD	AFFECTED REELS/TIME OF ACTION
First	The first reel of the file is to be acted upon at the time the file is opened.
Second	Each reel of the file, except the first, is to be acted upon at the beginning of the reel.
Third	Each reel of the file, except the last, is to be acted upon at the end of the reel.
Fourth	The last reel of the file is to be acted upon at the time the file is closed.

If this entry is included, all four operand fields must appear. They must not be separated by commas.

If the file is a single-reel file, the second and third operand fields should contain N.

If this entry is omitted, and the file is a tape file, the IOCS rewinds each reel of the file at the beginning and end of each reel.

An example of how this entry is coded is shown in Figure 33.

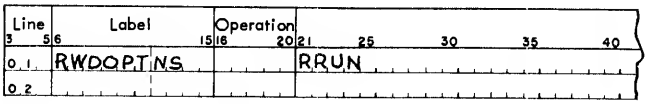


Figure 33. RWDOPTNS Entry

LABEL

This entry is not required. It is relevant only if the file is a tape file that uses labels. The LABEL entry provides the IOCS with the following information:

1. Whether the file uses 80-character or 120-character labels.
2. Whether the labels used by the file are standard (i.e., 1410 80-character or IBM Standard 120-Character labels) or nonstandard.
3. The file serial number, file identification, retention period, creation date, and reel sequence number of the first reel of the file, if the file uses standard labels.

The operands and operand fields of the LABEL entry are as follows:

OPERAND FIELD	OPERAND
First	80, if the file uses 80-character labels; 120 if the file uses 120-character labels.
Second	The five-digit file serial number, if the file uses standard tape labels; NONSTD, if the file uses nonstandard labels. (If NONSTD is entered here, the rest of the operand fields of this entry that follow this field are left blank.
Third	The file identification. If fewer than ten characters are entered, the IOCS left-justifies these characters and pads with blanks to obtain a ten-character file identification.
Fourth	The three-digit (80-character labels) or the four-digit (120-character labels) retention period.
Fifth	The five-digit creation date.
Sixth	The three-digit (80-character labels), or the four-digit (120-character labels) reel sequence number.

If the file uses standard labels all six of the operand fields discussed must appear in order. If the programmer does not wish to enter the indicated operand in a particular operand field, he may omit that operand. However, the comma that would normally separate the omitted operand from the next operand (if any) must be included. An example of how this entry is written is shown in Figure 34.

CHECK

This entry is required only if the file uses labels and the IOCS is to check all or part of these labels. This entry indicates to the IOCS what portions of these labels are to be checked. If this entry is omitted, the IOCS does not check any portion of the labels used by the file. The operands, the label fields they cause the IOCS to check, and the file type and label type affected by these operands are as follows:

OPERAND	LABEL FIELD	FILE TYPE	LABEL TYPE	METHOD OF CHECKING
SER	File serial number	Input	Header	Direct Compare*
ID	File identification	Input	Header	Direct Compare*
SEQ	Reel sequence number	Input	Header	Direct Compare*
DAT	Creation date	Input	Header	Direct Compare*
RET	Retention period	Output	Header	Old header creation date + retention period < current date
CNT	Block count	Input	Trailer	Direct Compare*
ALL	All the above			

* Information provided by DTF LABEL entry, updated by the IOCS where applicable, is compared against information read from the relevant label by the IOCS.

These operands can be entered in any order. If the operands RET or ALL are omitted, output file header labels are not read before creation of new header labels, and the reel serial number fields in the new output header labels contain blanks. The reason for this is that the reel serial number for new output headers are taken from the old output headers, which, in this case are not read. An example of how this entry is coded is shown in Figure 35.

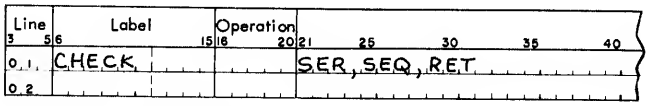


Figure 35. CHECK Entry

Sample DTF Statement

The basic DTF statement for a representative tape file (i.e., a file that consists of Form 2 data records, uses

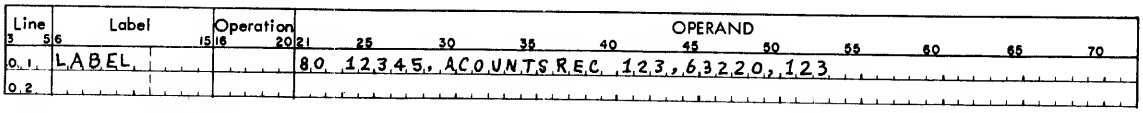
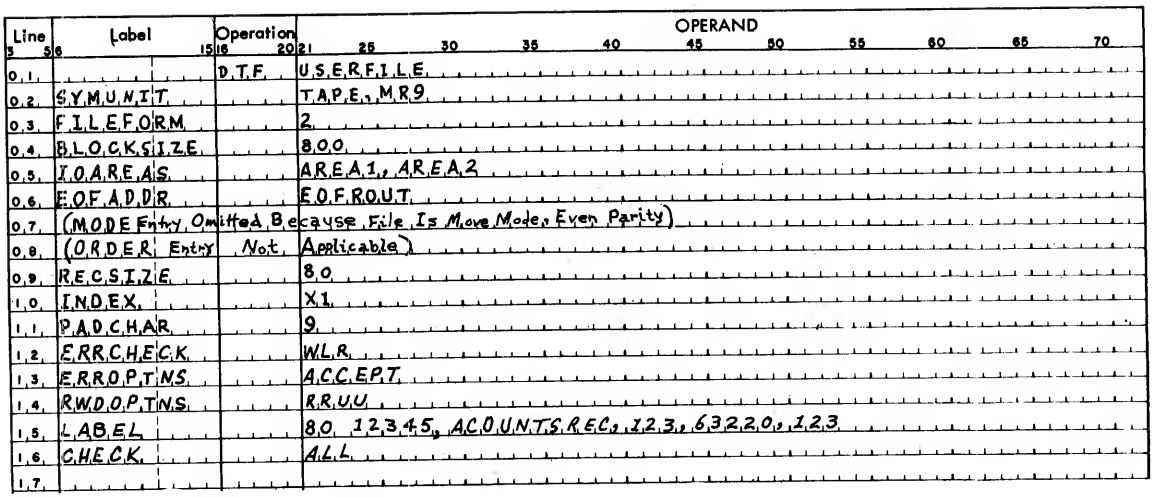


Figure 34. LABEL Entry



Line	Label	Operation	OPERAND
0.1		D.T.F.	U.S.E.R.F.I.L.E.
0.2	S.Y.M.U.N.I.T.		T.A.P.E.,.M.R.9.
0.3	F.I.L.E.F.O.R.M.		2.
0.4	B.L.O.C.K.S.I.Z.E.		8.0.0.
0.5	I.O.A.R.E.A.S.		A.R.E.A.1,.A.R.E.A.2
0.6	E.O.F.A.D.D.R.		E.O.F.R.O.U.T.
0.7	(M.O.D.E Entry Omitted Because File Is Move Mode, Even Parity)		
0.8	(O.R.D.E.R. Entry	Not	Applicable)
0.9	R.E.C.S.I.Z.E.		8.0
1.0	I.N.D.E.X.		X.1.
1.1	P.A.D.C.H.A.R.		9.
1.2	E.R.R.C.H.E.C.K.		W.L.R.
1.3	E.R.R.O.P.T.N.S.		A.C.C.E.P.T.
1.4	R.W.D.O.P.T.N.S.		R.R.U.U.
1.5	L.A.B.E.L.		8.0,.1.2.3.4.5,.A.C.O.U.N.T.S.R.E.C,.1.2.3,.6.3.2.2.0,.1.2.3.
1.6	C.H.E.C.K.		A.L.L.
1.7			

Figure 36. Representative DTF Statement

1410 80-character labels, and is read or written in Move mode, even parity) is shown in Figure 36.

IOCS Macro-Instructions

The IOCS macro-instructions are Autocoder statements entered in the user's source program. When these macro-instructions are encountered by the Autocoder processor, they cause the processor to produce 1410 or 7010 machine-language instructions in the user's object program. These machine-language instructions cause the IOCS to perform one or more of the functions of which it is capable.

The IOCS macro-instructions may be divided into those which cause the IOCS to perform the basic functions discussed in the introduction to this publication and those which cause the IOCS to perform additional or extended functions. The basic IOCS macro-instructions are covered in the material that follows. The extended IOCS macro-instructions are discussed under "Extended Use of the IOCS."

NOTE: All IOCS macro-instructions destroy the status of the High, Low, Equal and Zero Balance Indicators, but do not affect the arithmetic overflow or divide overflow indicators.

Basic IOCS Macro-Instructions

The basic IOCS macro-instructions are:

IOCTL OPEN, INPUT (Open Input File)
IOCTL OPEN, OUTPUT (Open Output File)
GET FILE (Make Next Logical Record Available)
GET FILE TO WORK (Make Next Logical Record Available and Move to WORK)
GET FILE TO FILE (Make Next Logical Record Available and Move to Output Area)
PUT FILE (Write Logical Record)
PUT WORK TO FILE (Move from WORK and Write Logical Record)
PUT FILE TO FILE (Move from Input Area and Write Logical Record)
IOCTL CLOSE, INPUT (Close Input File)
IOCTL CLOSE, OUTPUT (Close Output File)

General Functions

Each of the basic IOCS macro-instructions causes the IOCS to perform one or more general functions. These functions are discussed in the material that follows.

IOCTL OPEN

The IOCTL OPEN macro-instructions cause the IOCS to perform the following functions:

1. Link with the System Monitor to assign input and output files to the physical units specified for those files in the appropriate DTF SYMUNIT entries and ASGN cards. (The ASGN card is discussed in the *System Monitor* publication.)
2. Rewind, rewind and unload, backspace, or take no action on the first reel of tape input and output files, according to the information supplied by the relevant DTF RWDOPTNS entries (if any).
3. Read and check the header labels of tape input and output files, according to the information supplied by the appropriate DTF LABEL and CHECK entries.
4. Make the first output area associated with output files available to the using program, and place the high-order address of the output areas made available in the index registers specified by the relevant DTF INDEX entries (if any).
5. Force the first GET macro-instruction issued against input files to fill the first input area associated with those files, and schedule read operations to fill any additional input areas associated with those files.

GET

The GET macro-instructions cause the IOCS to perform the following functions:

1. Make the next logical record of the indicated file available to the using program in the area specified; deblocking the record when necessary.
2. Schedule read operations to fill those input areas associated with the indicated files (if any) that have been processed by the using program.
3. Begin processing any end-of-reel condition encountered on the indicated file.
4. Cause an exit to be taken to the end-of-file routine provided for the indicated file by the user, if the IOCS determines that an end-of-file condition exists on that file.

PUT

The PUT macro-instructions cause the IOCS to perform the following functions:

1. Prepare the current output area of the affected file to receive the next logical record of the file.
2. Block data records in the current output area of the affected file.
3. Schedule output operations that write on the affected file the data record(s) in the current output area of that file.
4. Begin processing end-of-reel conditions encountered on the affected file by branching to the IOCS Output End-Of-Reel routine.
5. Exit to the user's end-of-file routine for the affected file, if that file is a disk file and the IOCS determines that an end-of-file condition exists on that file.

IOCTL CLOSE

The IOCTL CLOSE macro-instructions cause the IOCS to perform the following functions:

1. Ensure completion of all requests pending for input/output operations against input and output files.
2. Pad incomplete blocks of Form 2 records if the affected files are output files.
3. Write all remaining records or blocks of records (i.e., those records or blocks that have not already been written) from the output areas of output files.
4. Write a tape mark on tape output files following the last data record or block of data records on those files.
5. Assemble and write trailer labels for tape output files that include a DTF LABEL or EXTEN entry in the DTF statements used to define those files.
6. Rewind, rewind and unload, backspace, or take no action on the last reel of tape input and output files, according to the information supplied by the appropriate DTF RWDOPTNS entries (if any).

Using the Basic Macro-Instructions

Figure 37 illustrates the use of IOCTL OPEN, GET, PUT, and IOCTL CLOSE.

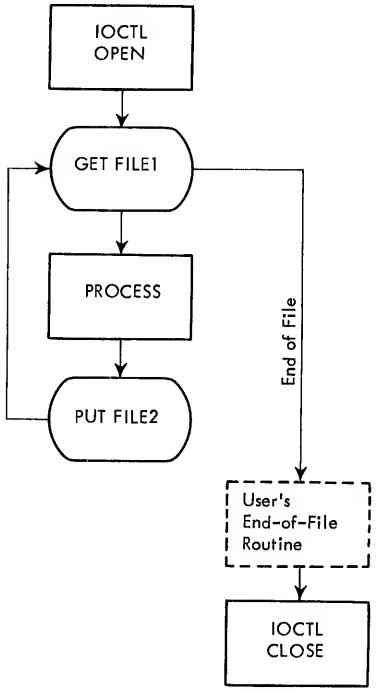


Figure 37. Basic Macro-Instructions

Coding the Basic Macro-Instructions

The basic IOCS macro-instructions are coded as noted in the paragraphs that follow.

IOCTL OPEN, INPUT

This macro-instruction is used to open input data files. The names of the input files that are to be opened appear as the operands of this macro-intruction. A maximum of eight files may be opened by each IOCTL OPEN,INPUT macro-instruction. An example of how this macro-instruction is coded is shown in Figure 38.

Line	Label	Operation	OPERAND
0.1	ANYLABEL	IOCTL	OPEN,INPUT,FILE1,FILE2,etc.
0.2			

Figure 38. IOCTL OPEN, INPUT

IOCTL OPEN, OUTPUT

This macro-instruction is used to open output data files. The names of the output files that are to be opened appear as the operands of this macro-instruction. A maximum of eight files may be opened by

each IOCTL OPEN, OUTPUT macro-instruction. Figure 39 shows an example of how this macro-instruction is coded.

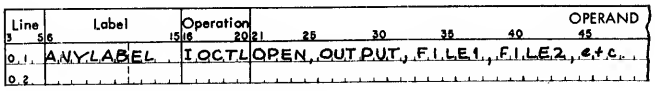


Figure 39. IOCTL OPEN, OUTPUT

GET FILE

The GET FILE macro-instruction makes the next logical record from the affected file (e.g., the file labeled INFILE in Figure 40) available to the using program. If a single input area has been specified for the file, the record is made available in that area. If multiple input areas are specified, the record is made available in the area with which the IOCS is currently working. An example of how this macro-instruction is coded is shown in Figure 40.

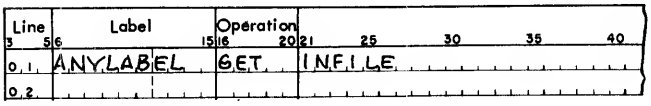


Figure 40. GET FILE

GET FILE TO WORK

The GET FILE TO WORK macro-instruction, in addition to the functions described under "GET FILE," causes the next logical record to be moved into an area other than an input area associated with the file (e.g., the area labeled WORK in the example of how this macro-instruction is coded, shown in Figure 41).

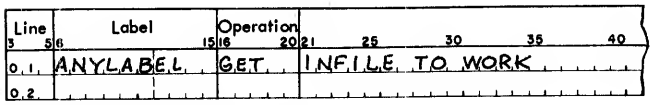


Figure 41. GET FILE TO WORK

GET FILE TO FILE

The GET FILE TO FILE macro-instruction, in addition to the functions described under "GET FILE," causes the next logical record to be moved into an output area. If a single output area is specified for the affected file (e.g., the file labeled OUTFILE in Figure 42), the record is moved into that area. If multiple output areas are specified, the record is moved into the area with which the IOCS is currently working.

To use GET FILE TO FILE, the programmer must specify the DTF INDEX entry for the affected output file, if the affected output file uses multiple output areas, or if the IOCS is to build blocks of records in the output area(s) of that file.

If the affected input and output files consist of Form 4 data records, the next record is not immediately moved into the current output area. Under these conditions the IOCS first checks the current output area of the affected output file. If the area contains enough positions to absorb the record, the IOCS then moves the record into the output area. If there are not enough positions, the contents of the output area are written on the affected output file, and the data record is moved into the output area as the first record of a new block.

In either case, a PUT FILE must be issued against OUTFILE to cause the IOCS: (a) to account for the record just moved into the output area, and (b) to select the next available portion of the output area.

NOTE: If this macro-instruction is used with two files; one that consists of Form 3 records and one that consists of Form 4 records: (a) the DTF RECSIZE entry for the Form 4 file must have the character 4 as its operand, and (b) the input area(s) associated with the Form 3 file must have a word mark over its high-order position.

An example of how this macro-instruction is coded is shown in Figure 42.

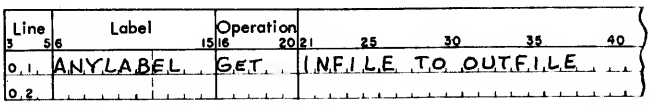


Figure 42. GET FILE TO FILE

PUT FILE

PUT FILE causes the IOCS to write a data record, or a complete block of data records, on the affected file (e.g., the file labeled OUTFILE in Figure 43). If a single output area was specified for the file, the record or block is written from that area. If multiple output areas were specified, the record or block is written from the output area with which the IOCS is currently working. This macro-instruction can be used with files consisting of Form 4 records only when used in conjunction with PUT FILE, FORM4. (See "PUT FILE, FORM4," under "Extended Use of the IOCS.") An example of how this macro-instruction is coded is shown in Figure 43.

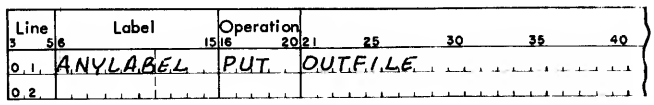


Figure 43. PUT FILE

PUT WORK TO FILE

PUT WORK TO FILE, in addition to the functions described under "PUT FILE," causes a data record in an area other than an input/output area (e.g., the area labeled WORK in Figure 44) to be moved to the current output area of the affected file (e.g., the file labeled OUTFILE). An example of how this macro-instruction is coded, is shown in Figure 44.

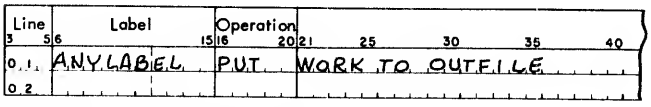


Figure 44. PUT WORK TO FILE

PUT FILE TO FILE

PUT FILE TO FILE, in addition to the functions described under "PUT FILE," causes the next logical record in the current input area of the affected input file (e.g., the file labeled INFILE in Figure 45) to be moved into the current output area of the affected output file (e.g., the file labeled OUTFILE in Figure 45).

NOTE: If this macro-instruction is used with two files; one that consists of Form 3 records and one that consists of Form 4 records: (a) the DTF RECSIZE entry for the Form 4 file must have the character 4 as its operand, and (b) the input/output area(s) associated with the Form 3 file must have a word mark over its high-order position.

To use PUT FILE TO FILE, the programmer must specify the DTF INDEX entry for the affected input file, if that file uses multiple input areas or consists of blocked records.

An example of how this macro-instruction is coded is shown in Figure 45.

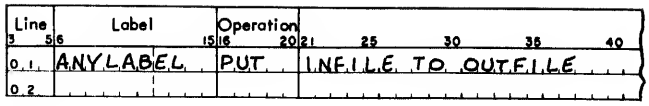


Figure 45. PUT FILE TO FILE

IOCTL CLOSE, INPUT

This macro-instruction is used to close input data files. The names of the input files that are to be closed appear as the operands of this macro-instruction. A maximum of eight files may be closed by each IOCTL CLOSE,INPUT macro-instruction. Figure 46 shows an example of how the macro-instruction is coded.

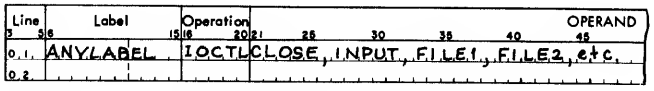


Figure 46. IOCTL CLOSE, INPUT

IOCTL CLOSE, OUTPUT

This macro-instruction is used to close output data files. The names of the output files that are to be closed appear as the operands of this macro-instruction. A maximum of eight files may be closed by each IOCTL CLOSE, OUTPUT macro-instruction. Figure 47 shows an example of how this macro-instruction is coded.

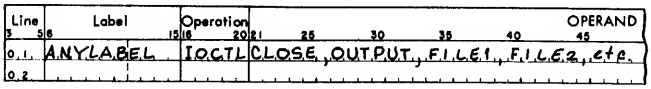


Figure 47. IOCTL CLOSE, OUTPUT

NOTE: No GET or PUT macro-instruction should be issued against a file before the file is opened or after the file is closed by the appropriate IOCTL macro-instruction.

This section provides the information necessary to make use of the additional or extended capabilities of the IOCS. These capabilities enable the IOCS to:

1. Use additional input/output areas to process a file. (See the DTF IOAREAS, and FILELIST entries.)
2. Provide exits to user-written error, service and tape label routines. (See, respectively, the DTF ERRADDR, INTADDR, LINE1, LINE2 and LINE1 ABOVE entries, and the discussions of user-written error, service, and tape label routines.)
3. Allow one File Table Extension to be used for more than one file. (See the DTF EXTEN entry.)
4. Provide deferred forms of the GET and PUT macro-instructions. (See GET FILE, DEFER and PUT FILE, DEFER.)
5. Provide an additional method of writing blocked, variable-length records on output files. (See the DTF LENGTH entry and PUT FILE, FORM4.)
6. Alter the stacker into which punched cards are to be selected during execution of the object program. (See PUT FILE,d; PUT WORK TO FILE,d; and PUT FILE TO FILE,d.)
7. Release partially processed blocks of data records. (See IOCTL RELSE.)
8. Force end-of-reel conditions. (See IOCTL FEOR.)
9. Write checkpoints. (See IOCTL CHKPT.)
10. Write specified areas of core storage on the console printer. (See IOCTL TYPE, AREA and IOCTL TYPE, AREA, DEFER.)
11. Perform certain input/output unit control functions, i.e., the rewinding or rewinding and unloading of tape reels. (See the UNCTL macro-instructions.)

Extended DTF Entries

The extended DTF entries may be broken down into basic entries with additional or extended operands and purely extended entries. These entries are listed below and discussed in the material that follows.

BASIC PLUS ADDITIONAL OPERAND(S)	EXTENDED
MODE	ERRADDR
ORDER	LENGTH
IOAREAS	INTADDR
ERRCHECK	LINE1
	LINE2
	LINE1 ABOVE
	EXTEN
	FILELIST

MODE

In addition to the operands discussed previously in this publication, a third operand may be included in this entry. This operand may be any Autocoder data move instruction mnemonic (e.g., MRCG). Figure 48 shows an example of how this entry is coded.

If the file is a Load mode file, all data move instructions generated as part of GET or PUT macro-instruction calling sequences (i.e., the coding generated by a GET or a PUT) that reference the file, are normally assembled as MRCWM move instructions. If the file is a Move mode file, all such data move instructions are normally assembled as MRCM move instructions. If the third operand of the MODE entry is included, the Autocoder mnemonic entered as this operand is substituted for the move instruction normally assembled (i.e., MRCWM or MRCM).

If the third operand is included and one or both of the other operands (i.e., LOAD and ODD) are omitted, the comma(s) that would normally separate the omitted operand(s) from the next operand must be retained. (See Figures 49 and 50.)

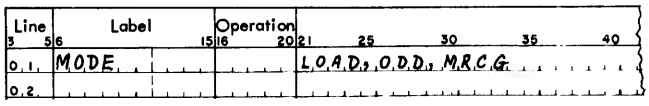


Figure 48. MODE Entry (Third Operand)

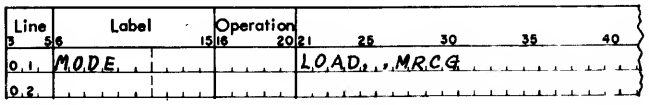


Figure 49. MODE Entry (ODD Omitted)

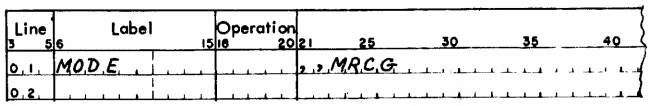


Figure 50. MODE Entry (LOAD and ODD Omitted)

ORDER

The following characters, in addition to those discussed previously, may appear as the operand of this entry:

OPERAND	EXPLANATION
9	The file is an input card file and the stacker into which cards read from the file are to be selected is specified during execution of the program by the UNCTL FILE,SSF,d macro-instruction. (If this operand is used, only one input/output area may be specified in the DTF IOAREAS entry.)
7	The file is a disk file and the Write Format Track instruction is to be generated for the file. NOTE: The following entries are not specifically prohibited. However, it should be noted that their usefulness is limited.
4	The file is a 1301 disk file and the Prevent Seek Complete instruction is to be generated for the file.
8	The file is a disk file and the Set Access Inoperative instruction is to be generated for the file.
9	The file is a disk file and the Release instruction is to be generated for the file.

The character entered as the operand of the ORDER entry becomes the low-order character of the x-control field of all the input/output instructions generated for the file. These input/output instructions are contained in the IORW's (Input/Output Request Words) that are used to process the file. (IORW's are discussed under "Internal Operation of the IOCS.")

An example of how this entry is coded is shown in Figure 51.

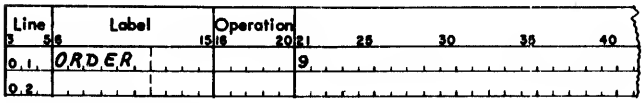


Figure 51. ORDER Entry

IOAREAS

The IOAREAS entry causes the IOCS to generate an Input/Output Request Word (IORW) for each input/output area whose label appears as an operand of the entry. The IOCS uses the IORW's generated to schedule and implement input/output operations on the file. If the programmer plans to generate these IORW's, this entry is not required. (Input/Output Request Words are discussed under "Internal Operation of the IOCS.")

The IOAREAS entry may have a fourth operand that was not covered in the previous discussion of the IOAREAS entry (see Figure 52). The fourth operand may be any label that consists of from one to nine alphameric characters. The IOCS takes this label, adds the character A, B, or C, and assigns the modified label, respectively, to the first, second (if any), and third (if any) IORW generated by the other operands of this entry.

If less than three input/output areas are specified by this entry, and labeling of the IORW's that are generated is desired, the comma(s) that would normally separate the labels entered as the second and/or third operands must be included in the entry (see Figure 53).

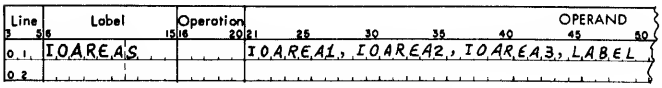


Figure 52. IOAREAS Entry (Four Operands)

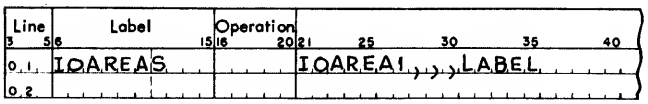


Figure 53. IOAREAS Entry (Two Operands Omitted, Labeling Desired)

ERRCHECK

In addition to the operands discussed earlier, this entry has two other possible operands (see Figure 55). They are:

OPERAND	EXPLANATION
STORE	This operand causes the IOCS to store the contents of the E- or F-address register (whichever is appropriate) in the Address Register Field of the affected IORW after execution of the input/output instruction in the Input/Output Operation Field of the IORW. (IORW's are discussed under "Internal Operation of the IOCS.") This operand should not be specified if the file consists of variable-length records, and the WLR operand has been specified for the file. Under these conditions, the IOCS places the length of the last record read from the file in the Address Register Field of the relevant IORW, instead of the contents of the E- or F-address register.
x	This operand can be any single character. If this operand (hereafter referred to as the fourth operand) is included, it must always be the last operand of the ERRCHECK entry. This operand is used to prevent the IOCS from checking, after each input/output operation performed on the file, those channel status indicators whose associated BCD bits are not included in the BCD code of the character entered. Extreme caution must be expected when using the operand. The channel status indicators and their associated BCD bits are shown in Figure 54.

NOTE: The IOCS does not consider the Busy channel status indicator (i.e., 2 bit) an error indicator. If this indicator is ON, the IOCS does not indicate that an error condition exists on the relevant file. Instead, the IOCS immediately retries the affected input/output operation.

Chonnel Stotus Indicotor	Associoted BCD Bit
B	Wrong-length record
A	No Tronsfer
8	Condition
4	Doto Check
1	Not Reody

Figure 54. Channel Status Indicators and Associated BCD Bits

If the fourth operand is omitted, the WLR operand is included, and the file consists of Form 1 or Form 2 records, the IOCS checks all the channel status indicators. If this operand and the WLR operand are omitted, regardless of the record form used by the file, the IOCS checks all the channel status indicators except the Wrong-Length-Record indicator.

If the fourth operand is used, its BCD code does not include the B bit, and the file consists of Form 1 or Form 2 records, the IOCS does not interrogate the Wrong-Length-Record indicator, even if the WLR operand has been specified. If the file consists of Form 3 or Form 4 records and the WLR operand was specified, the IOCS makes a program check for wrong-length records by comparing the length of the record just read or written with the relevant Block Character-Count regardless of whether the fourth operand is used and whether it contains a B bit.

If the fourth operand is used and any of the other operands are omitted (i.e., WLR, WDC, or STORE), the comma that would normally separate the omitted operand from the next succeeding operand must be included (see Figure 56).

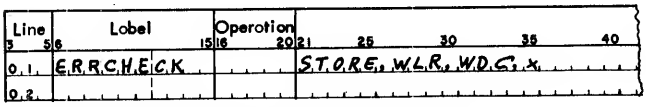


Figure 55. ERRCHECK Entry (Four Operands)

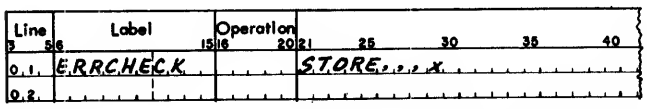


Figure 56. ERRCHECK Entry (Two Operands Omitted; Fourth Included.

ERRADDR

This entry is not required. It is used to specify the conditions under which the IOCS is to exit to an error routine the programmer has provided for the file.

The ERRADDR entry may have a minimum of one, and a maximum of six operands. The first operand is the label of the error routine the programmer has provided for the file. (This operand may be omitted. If it is, the comma that would normally separate it from the next operand must be included in the entry.) Each of the second through sixth operands consists of a single character. The characters that may appear as these operands are shown in Figure 57.

Filetype	Chorocter	Chonnel Stotus Indicotor(s)
Tape	1	Not Reody
	4	Doto Check
	8	Condition (End-of-File)
	-	Wrong Length Record
	M	Doto Check ond Wrong Length Record
	Q	Condition ond Wrong Length Record
	*	Doto Check, Condition, ond Wrong Length Record
	@	Dqto Check ond Condition
Disk	1	Not Reody (i.e., 7631 File Control Off-Line)
	9	Condition (i.e., 1301 Disk Storage Circuit Check, Involid Operotion code, or Write Disk Check Without Mode setting)
	5	Doto Check (i.e., Pority Check)
	V	Doto Check, No Tronsfer (i.e., Involid Trock Number), or Doto Check, No Tronsfer, ond Cond (i.e., Mode Check)
	Z	No Tronsfer, Condition (i.e., No Record Found, 1301 Disk Storoge Circuit Check, or 7631 File Control Circuit Check)
	4	Doto Check
	M	Wrong Length Record, Doto Check
	8	Condition (i.e., 1301 Disk Storoge Circuit Check)
	Q	Wrong Length Record, Condition (i.e., 1301 Disk Storoge Circuit Check plus Wrong Length Record)
	&	No Tronsfer (Write Inhibit Switch on 7631 is ON)
	-	Wrong Length Record (i.e., Incorrect Formot Length, or No GM-WM ot End of Disk Control Word)

Figure 57. Acceptable Characters

If: (a) the channel status indicator(s) turned on by an error condition, or conditions, that occurred on the file *exactly match* the channel status indicator(s) indicated by the BCD code of one of the single-character operands, and (b) two commas *do not* appear between the first operand and the single-character operand (see the second and third operands shown in Figure 58), the IOCS performs two operations. It bypasses its normal error procedures and causes a branch to be executed to the user's error routine. (If the first operand has been omitted, the IOCS bypasses its normal error correction procedures, and accepts the record.)

NOTE: This is the only type of exit that may be specified for files that reference unit-record devices.

If: (a) the channel status indicator(s) turned on by an error condition, or conditions, *exactly match* the channel status indicator(s) indicated by the BCD code of one of the single-character operands, and (b) two commas *do* appear between the first operand and that single-character operand (see the fourth, fifth, and sixth operands shown in Figure 58), the IOCS causes a branch to be executed to the user's error routine only if the normal IOCS error procedures have been executed, and the condition or conditions could not

be corrected. (If the first operand has been omitted, no branch is executed and the relevant operands become redundant.)

Any single-character operand of this entry whose BCD code includes the B bit, cannot have its intended effect, if: (a) the file consists of Form 1 or Form 2 data records, and (b) neither the WLR operand nor a fourth operand whose BCD code includes the B bit have been specified in the DTF ERRCHECK entry for the file. If the file: (a) consists of Form 3 or Form 4 data records, and (b) the WLR operand has been specified in the ERRCHECK entry for the file, any single-character operand of the DTF ERRADDR entry whose BCD code includes the B bit will have its intended effect, even though the BCD code of the fourth operand specified in the ERRCHECK entry for the file does not include the B bit.

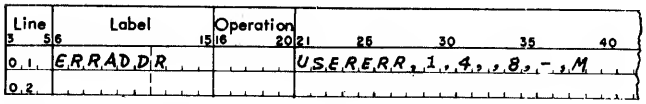


Figure 58. ERRADDR Entry

Disk Files: A group mark entered as the second or sixth operand of the DTF ERRADDR entry for a disk file causes the IOCS to exit to the user's error routine if *any* channel status indicator is turned on as the result of an input/output operation performed on the file.

If the group mark is entered as the *second* operand, the normal IOCS error procedures are bypassed, and any subsequent operands are redundant. (See Figure 59.)

If the group mark is entered as the *sixth* operand, the normal IOCS error procedures are not bypassed. In this case, any operand exeept the group mark operand entered after the double comma entry is superfluous. (See Figure 60.)

If the group mark appears as the sixth operand and the programmer wishes to omit one or more of the other operands, a comma must be substituted for each missing operand. (See Figure 61.)

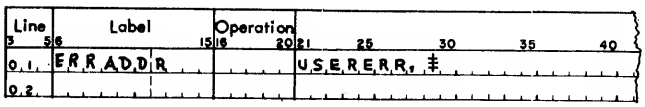


Figure 59. ERRADDR Entry (Second Operand; ‡)

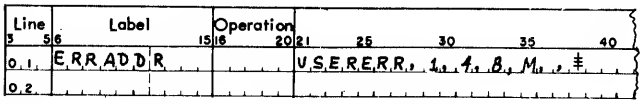


Figure 60. ERRADDR Entry (Sixth Operand; ‡)

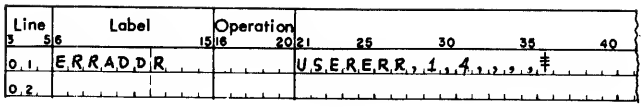


Figure 61. ERRADDR Entry (Operands Omitted)

In Figure 61, the first two commas entered before the group mark operand represent the omitted fourth and fifth operands. The third and fourth commas are the normal double-comma entry.

LENGTH

This entry is required if the file consists of Form 4 data records and the programmer plans to use the PUT FILE,FORM4 macro-instruction to write on the file. The operand of the LENGTH entry is a label the IOCS assigns to Field 12 of the File Table associated with the file. (This label must not be a linkage symbol. This label refers to the low-order position of the field.) Whenever the programmer is going to use the PUT FILE,FORM4 macro-instruction, he must first place, in this field, the estimated maximum length of the data record that is to be handled. The manner in which the IOCS utilizes this information is discussed under PUT FILE,FORM4. An example of how this entry is coded is shown in Figure 62. (See "The Extended IOCS Macro-Instructions" for a discussion of PUT FILE,FORM4 and "Internal Operation of the IOCS" for a description of the File Table.)

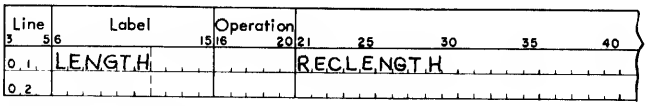


Figure 62. LENGTH Entry

INTADDR

This entry is not required. It is used to cause a branch to be taken to a service routine provided for the file by the programmer whenever execution of an input/output operation (other than a Seek Disk operation) is completed on the file.

The operand of the INTADDR entry is the label of the user-written service routine provided for the file.

If the DTF ERRADDR and INTADDR entries are both specified for the same file, the user's service routine is not entered if an input/output operation completed on the file results in an error condition, or conditions, that cause the IOCS to exit to the error routine the programmer has provided for the file.

If the SKIP operand of the DTF ERROPTNS entry and the INTADDR entry are specified for the same file, the

user's service routine is not entered if an input/output operation completed on the file results in an error condition, or conditions, that the IOCS error procedures cannot correct. An example of how this entry is written is shown in Figure 63. (User-written service routines are discussed under "User-Written Routines.")

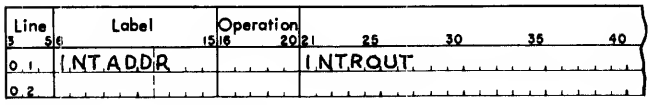


Figure 63. INTADDR Entry

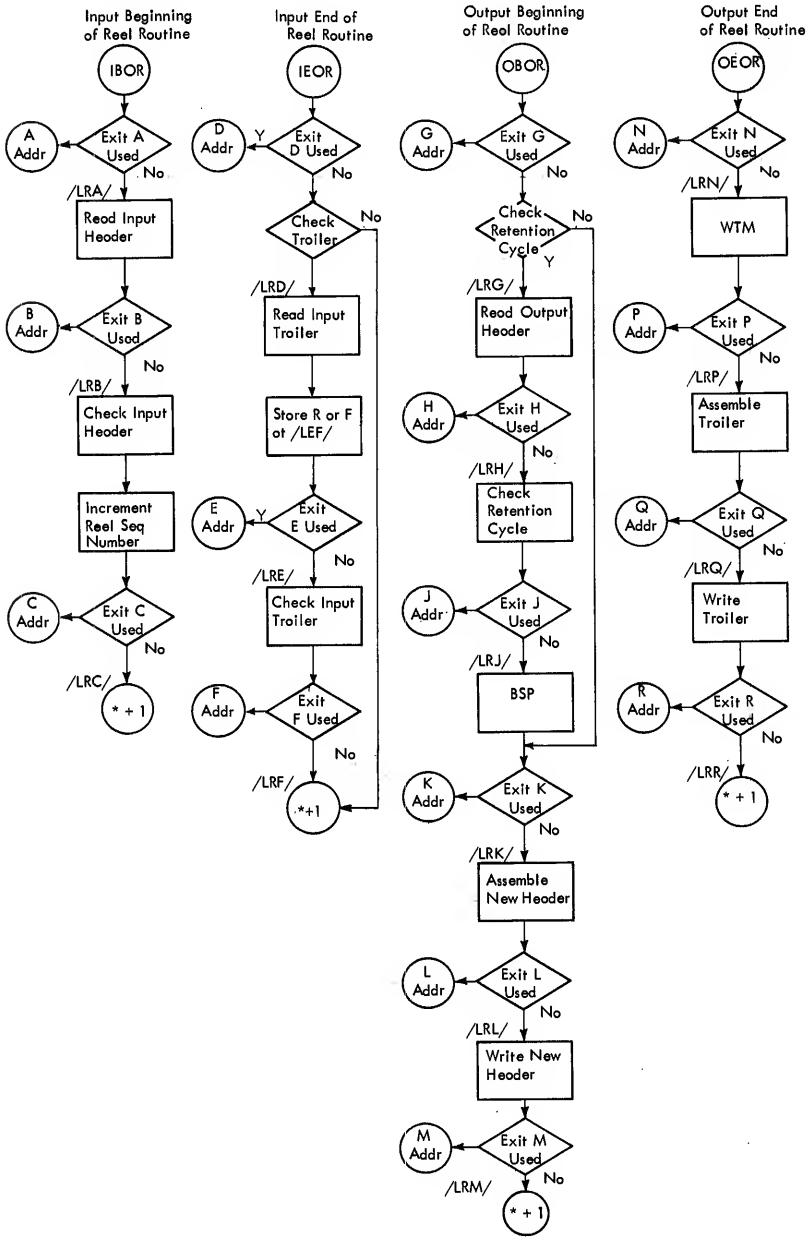


Figure 64. IOCS Tape Label Routines

LINE1

This entry is not required. It is of significance only if the file is a tape file that uses labels. This entry allows the programmer to specify up to four exits to tape label routines the user has provided for the file.

There are 16 exits (labeled A-H, J-N, P-R) available. These exits are in the IOCS tape label routines. The position of each of these exits within the various IOCS label routines is shown in Figure 64 (e.g., Exit A in the Input Beginning-of-Reel routine). The system symbols that identify the return points from these exits are also shown in Figure 64 (e.g., /LRA/; the return point from Exit A.)

The exits provided are taken depending on which reel of the file the relevant IOCS label routine is processing. The characters R, F, or B indicate which reels of the file are to be affected, as follows:

CHARACTER	REELS AFFECTED
R	All reels of the file except the first, if the IOCS tape label routine in which the exit is to be taken is an input routine. All reels of the file except the last if the affected IOCS label routine is an output routine.
F	Only the first reel of the file, if the affected IOCS label routine is an input routine. Only the last reel, if the affected IOCS label routine is an output routine.
B	All reels of the file, regardless of the IOCS label routine in which the exit is to be taken.

Each exit that is to be taken is coded as shown in Figure 65.

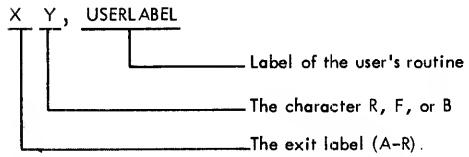


Figure 65. Coding Format for Exits to User-Written Label Routines (LINE1)

From one to four exits can be specified using this entry. The label LINE1 is entered in the proper field of the line of the coding sheet used for this entry and the relevant exits are specified on that same line. Each exit specified must be separated from the next by a comma. An example of how this entry is coded is shown in Figure 66.

LINE2

This entry cannot be used unless the LINE1 entry has been specified. This entry allows the programmer to specify from one to four additional exits to user-written tape label routines. Each exit is coded as described under the LINE1 entry, and must be separated from the next (if any) by a comma. An example of how this entry is coded is shown in Figure 67.

LINE1 ABOVE

This entry is used to specify exits from the IOCS tape label routines to tape label routines written for the file

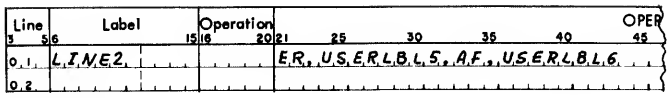


Figure 67. LINE2 Entry

by the programmer when more than eight such exits are desired. This entry is not required.

The LINE1 ABOVE entry cannot be included if the LINE1 entry has been specified for the file.

The desired exits are specified by providing coding of the format shown in Figure 68 for each exit. The exits and characters that can be included in this coding are the same as those discussed under the DTF LINE1 entry.

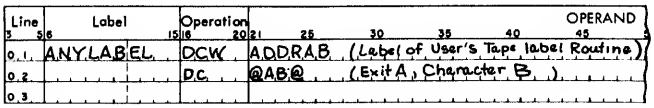


Figure 68. Coding Format for Exits to User-Written Label Routines (LINE1 ABOVE)

Once the desired exits are specified, the relevant coding must be placed before the DTF Header Line entry for the file, as shown in Figure 69, and must be preceded by a single-character DCW like that shown on Line 1 of Figure 69. (This character can be any character except a special character such as $.) To complete the entry, the exact coding shown on Line 13 of Figure 69 must be included among the DTF entries for the file.

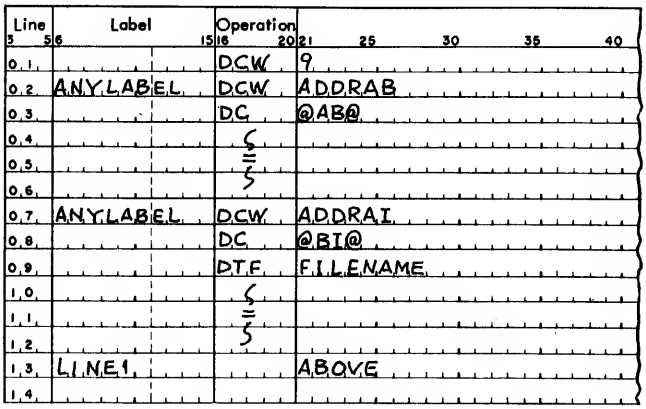


Figure 69. Placement of Coding for Additional Exits

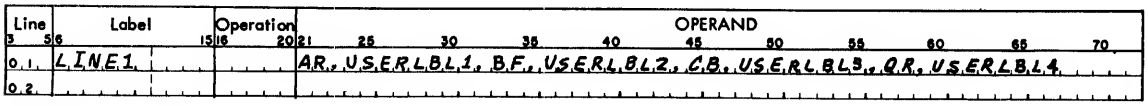


Figure 66. LINE1 Entry

EXTEN

This entry is not required. It applies only to tape files that use labels which the IOCS is to check. This entry can be used to label the File Table Extension generated for the file by the DTF LABEL entry. If the DTF LABEL entry has been omitted, the EXTEN entry must be used to indicate which File Table Extension the IOCS is to reference while processing the labels of the file. (The File Table Extension is discussed under "Internal Operation of the IOCS.")

The operand of the EXTEN entry is one of the following:

1. For files where the DTF LABEL entry has been specified, it is the label assigned to the File Table Extension generated by the DTF LABEL entry.
2. For files where the DTF LABEL entry has been omitted, it is the label that has been assigned to a File Table Extension the programmer generated, or a File Table Extension generated by the DTF LABEL entry specified for another file. An example of how this entry is coded is shown in Figure 70.

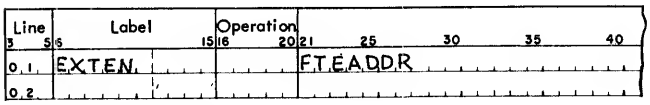


Figure 70. EXTEN Entry

FILELIST

This entry is used to associate IORW's the *programmer* has generated with a specific data file. The operand of the FILELIST entry is the low-order address of the Link Field of an IORW, or the first of a list of IORW's the programmer has created.

The IOCS places this address in Field 5 of the appropriate File Table, if the DTF IOAREAS entry has not been specified for the file. If the DTF IOAREAS entry has been specified, the IOCS places this address in the Link Field of the last IORW on the File List associated with the file. (File Tables and File Lists are discussed under "Internal Operation of the IOCS.") An example of how this entry is coded is shown in Figure 71.

If the FILELIST entry is used, the IOCS assumes that the file uses two or more input/output areas. For this reason, if any macro-instructions except GET FILE TO WORK, PUT WORK TO FILE, or PUT WORK TO FILE, d are issued against the file, the DTF INDEX entry must also be included in the DTF statement written for the file.

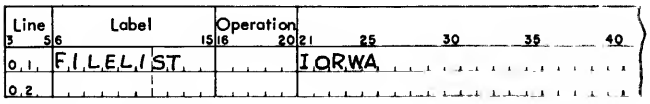


Figure 71. FILELIST Entry

User-Written Routines

Five types of user-written routines can be included in programs that incorporate the IOCS. The five types are end-of-file, error, service, scramble, and label routines. End-of-file, error, service and label routines are discussed here. Scramble routines are discussed under "Disk File Processing."

NOTE: The IOCS does not store or alter the status of the arithmetic overflow or divide overflow indicators before entering user-written routines.

USER-WRITTEN SERVICE ROUTINES

The IOCTL ENTRY macro-instruction must be the first instruction of every user-written service routine. The second operand of this macro-instruction must be the label entered as the operand of the DTF INTADDR entry specified for the file. The label that appears as the second operand of this macro-instruction refers to the low-order position of the coding (see Figure 72) developed by the macro-instruction. This coding immediately precedes the user-written routine. For this reason, the first instruction of this routine can be referred to by this label +1.

The IOCTL EXIT macro-instruction must be the last instruction of every user-written service routine. (The IOCTL ENTRY and IOCTL EXIT macro-instructions are discussed under "The Extended IOCS Macro-Instructions.")

User-written service routines must not issue a GET or PUT macro-instruction that results in detection of an

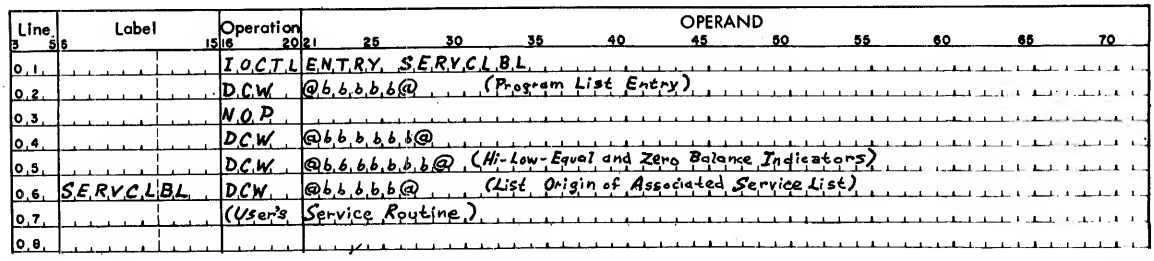


Figure 72. Coding Developed by IOCTL Entry

end-of-reel or end-of-file condition. User-written service routines must not issue IOCTL OPEN, CLOSE, RELSE, or FEOR at a time when the IOCS may be processing an IOCTL OPEN, CLOSE, RELSE, or FEOR encountered in the user's main-line program, unless a Tele-Processing device has been specified for the using installation at System Generation by means of the DEVDF macro-instruction. (See the *System Generation* publication.)

User-written service routines cannot issue a GET or PUT against the file with which they are associated, unless at least one IORW is on the File List of the file at the time the macro-instruction is issued. User-written service routines cannot issue a GET or PUT against a file other than the one with which they are associated if the DTF INTADDR entry has been specified for the affected file, unless there is at least one IORW on the File List of the affected file at the time the GET or PUT is issued. In a TP environment, Service routines should not use index registers for communication between themselves and other Service routines, or themselves and the main line.

USER-WRITTEN LABEL ROUTINES

The IOCS reads tape labels into and writes tape labels out of the areas shown in Figure 73. These areas are referred to by the system symbols indicated.

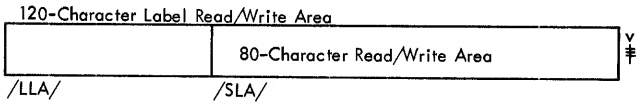


Figure 73. Tape Label Read/Write Areas

Except for IOCTL OPEN, CLOSE, RELSE and FEOR, user-written label routines can utilize all the IOCS macro-instructions, provided their use does not result in an end-of-reel or end-of-file condition (i.e., a GET that causes the final tape mark to be sensed, or a PUT that causes the final reflective strip to be sensed).

Generalized user-written label routines that process the labels of more than one file can obtain the address of the File Table of the affected file from index register 13, or from a five-digit field whose low-order address is referred to by the system symbol /LFA/. (AB zone bits appear over the units position of both index register 13 and the five-position field.)

Before returning to the Input End-of-Reel routine beyond Exit E (see Figure 64), user-written label routines designed to process nonstandard input trailer labels may place either "R" or "F" in a location referred to by the system symbol /LEF/. The character should be R if the programmer wishes the IOCS to process the next reel of the file; the character should be F if no additional reels of the file are to be processed.

USER-WRITTEN ERROR ROUTINES

At the time a branch is executed to a user-written error routine, the following information is available to the programmer:

1. Index register 13 contains the address of the affected File Table. (AB zone bits appear over the units position of X13.) However, the contents of X13 are destroyed if any IOCS macro-instruction is issued within the routine or a branch instruction is executed to the Resident Monitor.
2. Field 10 of the affected File Table contains the address of the re-entry point into the calling sequence generated by the GET or PUT macro-instruction, the execution of which uncovered the relevant error condition.
3. The affected IORW is at the top of the File List with which it is associated.
4. The Channel Status Character in the affected IORW indicates the channel status indicators that were turned on by the relevant error condition.

Once a branch has been executed to a user-written error routine, that routine can cause the erroneous record: (a) to be processed as if it were error free, (b) to be overlaid, if the file is an input file, by reading the next logical record from the affected file into the input area that contains the erroneous record, (c) to be backspaced over, if the file is a tape file, and reread into or rewritten out of current input/output area of the affected file, or (d) to be written on the system's Core Image file or another file set up by the programmer for this purpose.

If the erroneous record is to be processed as if it were free of error, the user-written error routine must cause a branch to be executed to the address contained in Field 10 of the affected File Table.

If the next logical record is to be read into the same area that contains the erroneous record, the user's routine must change the B bit in the second File Indicator (F2) of the affected File Table to an A bit by executing the instruction MRZR ,FILENAME+12, and cause a branch to be taken to the address in Field 10 of the affected File Table.

If the erroneous record is to be reread or rewritten, a UNCTL FILE,BSP macro-instruction must be issued against the relevant file, the block counter (Field 2) in the affected File Table must be decremented by one, and the steps outlined in the preceding paragraph must be executed.

If the erroneous record is to be written on the Core Image file, the programmer must have coded a DTF statement for that file, included MDM as the second operand of the relevant DTF SYMUNIT entry, included the DTF MODE entry with its ODD operand, and opened the

file by means of the IOCTL OPEN,OUTPUT macro-instruction. The user's routine must then place the B-address of the input/output operation that appears in the affected IORW in the IORW associated with the Core Image file, and issue a PUT macro-instruction against the Core Image file.

This action causes the erroneous record to be written on the Core Image file. The erroneous record may then be processed or overlaid as described in the preceding paragraphs.

If the erroneous record is to be written on a file set up by the programmer for this purpose, the procedures outlined for writing the erroneous record on the Core Image file must be followed, with one exception. The second operand of the relevant DTF SYMUNIT entry cannot be MDM.

NOTE 1: If a file consists of Form 4 records and the error routine provided for the file determines that a block of records on which an error condition exists is to be processed as if it were error free, the programmer must make sure that the Record Character-Count in each record of the block is accurate.

NOTE 2: User error routines provided for an output file cannot issue IOCTL OPEN, CLOSE, FEOR, OR RELSE.

USER-WRITTEN END-OF-FILE ROUTINES

Once the end-of-file routine provided for a file has been entered, no further GET or PUT macro-instructions can be issued against the affected file, unless the file is closed and then reopened. The work files discussed in Appendix E are an example of the type of files on which this situation might frequently occur.

When an end-of-file condition occurs on a disk output file, the IOCS removes all IORW's associated with the file from the Read/Write List and places them on the bottom of the File List associated with the file. The IOCS next causes a branch to be taken to the end-of-file routine the user has provided for the file. The programmer cannot then use the IOCTL CLOSE, OUTPUT macro-instruction to write on the file any data remaining in the output areas associated with the file. The reasons for this are: (a) the file cannot accept any additional data, and (b) the IORW's that refer to the relevant output areas are on the File List of the file. The programmer must himself determine where and how any data contained in these output areas is to be disposed of.

Extended IOCS Macro-Instructions

The extended IOCS macro-instructions are listed below, and discussed in the paragraphs that follow.

```
GET FILE,DEFER              IOCTL CHKPT
PUT FILE,DEFER              IOCTL TYPE,AREA
PUT FILE,FORM4              IOCTL TYPE,AREA,DEFER
PUT FILE,d                  UNCTL FILE,CC,d
PUT WORK TO FILE,d          UNCTL FILE,SSF,d
PUT FILE TO FILE,d          UNCTL FILE,BSP
IOCTL RELSE,(I/O),FILE      UNCTL FILE,RWD
IOCTL FEOR,(I/O),FILE       UNCTL FILE,RWU
IOCTL ENTRY                 UNCTL FILE,WTM
IOCTL EXIT
```

GET FILE,DEFER

This macro-instruction is designed for use with input files that consist of unblocked records and use a single input area. It allows the programmer to overlap read operations and processing when only one input area has been assigned to the file. This macro-instruction causes the IOCS to schedule execution of an input operation that is to read a record from the indicated file, (e.g., the file labeled INFILE in Figure 74) into the input area specified for the file. This macro-instruction then causes the IOCS to branch to the next sequential instruction of the using program without waiting for execution of the requested input operation and subsequent release of the input area to the using program (i.e., the IOCS defers making the requested record available in the input area of the file.)

Once GET FILE,DEFER has been issued, the using program can process any data except data contained in the input area. When the program has completed whatever processing is required, GET FILE; GET FILE TO WORK; or GET FILE TO FILE can be issued. These macro-instructions cause the IOCS to make the record read as a result of issuing GET FILE,DEFER available to the user's program.

If the programmer wishes, instead of GET FILE; GET FILE TO WORK; or GET FILE TO FILE, the character in Field Q4 of the affected IORW can be tested to determine if the relevant input area is available. If the BCD code of the character in Field Q4 includes the A bit, the requested input operation has been completed and the associated area is available. If the programmer substitutes a B bit for the A bit, he may access the contents of the input area. If the BCD code includes the B bit, the requested operation has not been completed and the programmer can: (a) process other data, (b) wait until the IOCS replaces the B bit with an A bit, or (c) issue one of the GET macro-instructions noted above, to effect completion of the operation.

If GET FILE,DEFER is used, the IOCS suppresses the check for end of file that the other GET macro-instructions cause it to make. However, the use of GET FILE, GET FILE TO WORK, or GET FILE TO FILE, subsequent to the use of GET FILE,DEFER causes the IOCS to recognize an end-of-file condition that results from the issuance of GET FILE,DEFER. An example of how this macro-instruction is coded is shown in Figure 74.

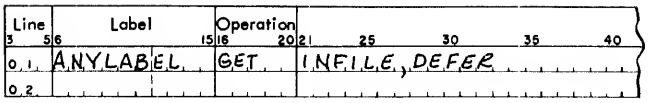


Figure 74. GET FILE,DEFER

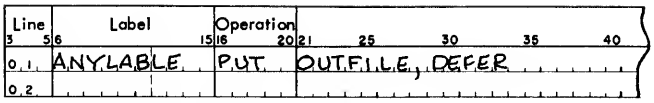


Figure 75. PUT FILE,DEFER

PUT FILE,DEFER

This macro-instruction is designed for output files that consist of unblocked records and use a single output area. It allows the programmer to overlap write operations and processing when only one output area has been assigned to the file. PUT FILE,DEFER causes the IOCS to schedule execution of an output operation that is to write, on the affected file (e.g., the file labeled OUTFILE in Figure 75), a record taken from the output area specified for the file. This macro-instruction then causes the IOCS to branch to the next sequential instruction of the user's program, without waiting for actual execution of the requested write operation and release of the output area to the program (i.e., the IOCS defers making the output area available to the using program).

Once PUT FILE,DEFER has been issued, the using program can process any data except that data contained in the output area of the file. When the program has finished processing the relevant data, PUT FILE can be issued. This macro-instruction causes the IOCS to ensure that the write operation scheduled by PUT FILE,DEFER has been executed. The IOCS then makes the output area of the file available to the program.

If the programmer wishes, instead of issuing PUT FILE the character in Field Q4 of the affected IORW can be tested to determine if the relevant output area is available. If the BCD code of the character in Field Q4 includes the A bit, the requested write operation has been executed, and the output area associated with the file is available to the using program. If the programmer substitutes a B bit for the A bit, he may access the output area.

If the BCD code of the character includes the B bit, the requested operation has not completed and the programmer can: (a) process other data, (b) wait until the IOCS replaces the B bit with an A bit, or (c) issue a PUT macro-instruction against the file to effect the completion of the requested write operation.

If PUT FILE,DEFER is used, the IOCS suppresses the check for end of file which other PUT macro-instructions cause it to make. However, the subsequent use of PUT FILE; PUT WORK TO FILE; OR PUT FILE TO FILE causes any end-of-file condition that resulted from use of PUT FILE,DEFER to be recognized by the IOCS. An example of the way in which this PUT macro-instruction is coded is shown in Figure 75.

PUT FILE,FORM4

This macro-instruction is only used when the affected file (e.g., the file labeled OUTFILE in Figure 76) consists of Form 4 records *and* the programmer plans to move the next logical record into the current output area of that file by means of a move instruction, rather than by means of an IOCS macro-instruction.

PUT FILE,FORM4 is used as follows:

1. The programmer places the actual length, or the maximum possible length, of the record that is to be moved in a five-digit field whose label is the operand of the DTF LENGTH entry specified for the file. This placement should be accomplished by means of a Zero and Add instruction (ZA).

2. The programmer issues PUT FILE,FORM4. This macro-instruction causes the IOCS to check the current output area of the file. (In performing this check, the IOCS destroys the contents of the five-digit field that is referred to by the label entered as the operand of the LENGTH entry.)

If the current area does not contain enough positions unoccupied by data to absorb the record, the current contents of the area are written on the file and the IOCS branches to the next sequential instruction of the program.

If the current area contains enough positions to absorb the record, the IOCS branches to the next sequential instruction of the program.

3. The programmer moves the next logical record into the current output area by means of a move instruction that wipes out any extraneous word marks in that portion of the current output area that is taken up by the next logical record.

4. The programmer issues PUT FILE. This macro-instruction causes the IOCS to account for the record the programmer moved into the output area and to branch to the next sequential instruction of the program. An example of how this PUT macro-instruction is coded is shown in Figure 76.

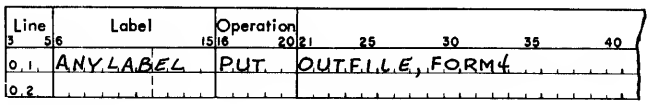


Figure 76. PUT FILE,FORM4

PUT FILE,d PUT WORK TO FILE,d PUT FILE TO FILE,d

These macro-instructions may only be used with punched card or printer files. They perform, respectively, the same functions as PUT FILE, PUT WORK TO FILE and PUT FILE TO FILE.

In addition, these macro-instructions:

1. For punch files, allow the programmer to alter the stacker assigned to the file during execution of the user's program. (This is accomplished by substituting 0, 4, or 8 for the d-character shown in Figure 77.)

2. For printer files, allow the programmer to specify, during execution of the program, whether the Write A Line or the Write Word Marks instruction is to be used to write on the file. This is acomplished by substituting 0 (i.e., Write A Line) or 1 (i.e., Write Word Marks) for the d-character shown in Figure 77.

The last PUT FILE,d; PUT WORK TO FILE,d; or PUT FILE TO FILE,d encountered during execution of the using program establishes the stacker or print instruction that is to be used for the balance of the program, provided the file uses a single output area. If the file uses multiple output areas, the programmer must issue PUT FILE,d; PUT WORK TO FILE,d; or PUT FILE TO FILE,d for the balance of the program whenever he issues a PUT macro-instruction against the affected file.

If none of these macro-instructions are used in the program, the stacker or instruction specified in the DTF ORDER entry for the file is the stacker or instruction used throughout the program. An example of how these macro-instructions are coded is shown in Figure 77.

Line	Label	Operation	
01	ANYLABEL	PUT	OUTFILE,d
02	ANYLABEL	PUT	WORK TO OUTFILE,d
03	ANYLABEL	PUT	INFILE TO OUTFILE,d

Figure 77. PUT FILE,d; PUT WORK TO FILE,d; and PUT FILE TO FILE,d

IOCTL RELSE,(INPUT/OUTPUT),FILE

This macro-instruction may only be used with files that consist of Form 2 or Form 4 data records. This macro-instruction causes the IOCS to make the current block of data records from the relevant file unavailable to the user's program. The first operand of this macro-instruction is RELSE (Release), the second operand is INPUT (for input files), or OUTPUT (for output files); the third operand is the name of the affected file.

If the second operand is INPUT, the IOCS ignores any records in the current input area that have not been made available to the program, and the next GET macro-instruction issued against the file causes the IOCS to make the first record of the next logical block of records available to the program.

If the second operand of this macro-instruction is OUTPUT, the IOCS: (a) ignores the fact that the block of records in the current output area of the file is incomplete, (b) pads the incomplete block if the file consists of Form 2 records, and (c) schedules the writing of the block on the appropriate file. The next logical record moved into the output area then becomes the first record of a new block. An example of how this macro-instruction is written is shown in Figure 78.

Line	Label	Operation	
01	ANYLABEL	IOCTL	RELSE,INPUT,FILENAME
02			

Figure 78. IOCTL RELSE,(INPUT/OUTPUT),FILE

IOCTL FEOR,(INPUT/OUTPUT),FILE

This macro-instruction is issued against tape files to force an end-of-reel condition. The first operand of this macro-instruction is FEOR (Force End of Reel). The second operand is INPUT (for input files) or OUTPUT (for output files). The third operand is the label of the affected file.

If the second operand is INPUT, a branch is immediately taken to the IOCS Input End-of-Reel routine. As a result, the trailer label of the affected reel is not processed, and no check is made to determine if an end-of-file condition exists on the file.

If the second operand of this macro-instruction is OUTPUT, the IOCS: (a) completes all output operations issued against the file that are still pending, (b) pads the last block of data records if the file consists of Form 2 records and the last block is incomplete, (c) writes on the file all records (or blocks of records) in the output area(s) of the file, and (d) branches to the appropriate IOCS Output End-of-Reel routine. An example of how this macro-instruction is written is shown in Figure 79.

Line	Label	Operation	
01	ANYLABEL	IOCTL	FEOR,OUTPUT,FILENAME
02			

Figure 79. IOCTL FEOR,(INPUT/OUTPUT),FILE

IOCTL ENTRY

The IOCTL ENTRY macro-instruction must be the first instruction of every user-written service routine. It is used to define the entry point into the user-written service routine whose label appears as its second

operand. Figure 80 shows an example of how this macro-instruction is written. (The coding generated by this macro-instruction is shown in Figure 72.)

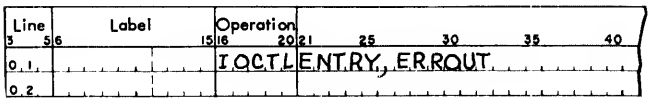


Figure 80. IOCTL ENTRY

IOCTL EXIT,ACCEPT IOCTL EXIT,DELETE

One of these macro-instructions must appear as the last instruction of every user-written service routine (see Figures 81 and 82). IOCTL EXIT,ACCEPT causes the IOCS to accept the results of the input/output operation, the completion of which initiated entry into the affected user-written service routine. IOCTL EXIT,DELETE causes the IOCS to note that the IORW associated with the input/output operation, the completion of which initiated entry into the affected user-written routine, has been deleted from the group of IORW's (i.e., the Service List) associated with the routine. The IOCS assumes the user has disposed of the IORW. (Service Lists are described under "Internal Operation of the IOCS.")

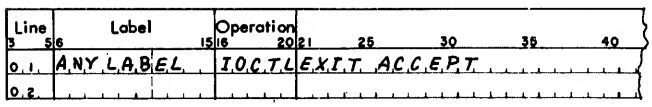


Figure 81. IOCTL EXIT,ACCEPT

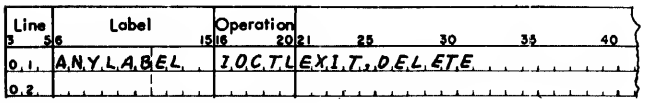


Figure 82. IOCTL EXIT,DELETE

IOCTL CHKPT

This macro-instruction causes the IOCS to write three records on the Core Image file. The first of these records includes the entire contents of core storage. The second includes the contents of core storage from the last position occupied by the Resident Monitor +1 to the end of core storage. The third includes the contents of core storage from the address specified as the tenth operand of the IOKDF macro-instruction to the end of core storage. (See the *System Generation* publication.) This address is automatically stored in a five-position field whose low-order position may be referred to by the system symbol /ORG/.

Once this macro-instruction has been executed a message is written on the console printer notifying the operator that a checkpoint is being taken.

The use of this macro-instruction allows the program to be restarted at the point in the program where the IOCTL CHKPT macro-instruction was issued. For a description of the procedure for restarting programs from a checkpoint see the publication, *IBM 1410/7010 Operating System; Operator's Guide,* Form C28-0351.

Programs that use the IOCTL CHKPT macro-instruction must observe the following conventions:

1. Each time a tape is backspaced by means of the UNCTL FILE,BSP macro-instruction, the block counter (Field 2 of the File Table) of the affected file must be decremented by one.

2. If any reel of a tape file has been rewound by means of the UNCTLFILE,RWD or UNCTLFILE,RWU macro-instruction since the last time the file was opened, that file must be closed before the IOCTL CHKPT macro-instruction is executed if the file uses labels. If the file does not use labels, it need not be closed, but Field 2 of the relevant File Table must be zeroed before the macro-instruction is executed.

3. The IOCTL CHKPT macro-instruction cannot be used if a file is open for which N has been specified in the operand field of the DTF RWDOPTNS entry, and the condition specified by the operand field in which it appears (e. g., beginning of reel, end of reel, etc.) has occurred.

4. If two or more files use the same tape unit, only one of these files can be open at the time the IOCTL CHKPT macro-instruction is executed.

5. If one or more IORW's have been added to the File List associated with a tape file after the file is opened, and the new IORW's reference a different tape unit than the IORW's that were on the File List at the time the file was opened, the file must be closed before the IOCTL CHKPT macro-instruction is executed.

If the IORW's on the File List of a tape file have been altered, since the file was opened, to reference a tape unit other than the unit they were assigned to at the time the file was opened, the affected file must be closed before the IOCTL CHKPT macro-instruction is executed.

6. Files which use 1410 80-character or IBM Standard 120-Character labels cannot be open at the time the IOCTL CHKPT macro-instruction is executed if the File Table Extension address or the contents of the File Table Extension used by the files have been modified since the file was opened.

7. Any file that uses nonstandard labels cannot be open at the time the IOCTL CHKPT macro-instruction is executed, if the exits provided in the IOCS tape label routines are used by the programmer to bypass IOCS reading or writing of the labels used by the file.

8. Word separator characters with word marks can-

not be present in that area of core storage occupied by the using program at the time the IOCTL CHKPT macro-instruction is executed.

9. Checkpoints cannot be taken at a point in the user's program that will require the restart program to position a reel of tape mounted on a 7330 magnetic tape unit at a record that is longer than the algebraic difference between the integers at /OGR/ and /ORG/.

After an IOCTL CHKPT macro-instruction has been executed, a three-position field (whose units position can be referred to by the system symbol /CPT/) contains the serial number (hundreds and tens positions of the field) of the checkpoint that was just written. The units position of this field contains an indicator that the using program can interrogate to determine if the program has been restarted from the last checkpoint taken. This indicator is the character R if the program has been restarted. In this case, the hundreds and tens positions indicate the serial number of the checkpoint from which the program was restarted. An example of how this entry is coded is shown in Figure 83.

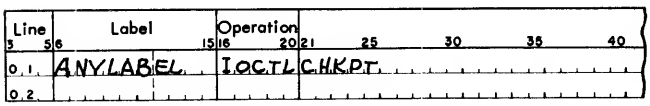


Figure 83. IOCTL CHKPT

IOCTL TYPE,AREA

This macro-instruction is used to cause the IOCS to write the contents of a specified area of core storage on the console printer. This action is taken before the next sequential instruction of the using program is executed. The first operand of this macro-instruction is TYPE. The second operand is any label that refers to the high-order position of the area whose contents are to be written on the console printer. A group mark with word mark must appear immediately to the right of the low-order position of this area. Figure 84 shows an example of how this macro-instruction is coded.

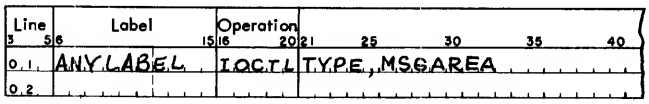


Figure 84. IOCTL TYPE,AREA

IOCTL TYPE,AREA,DEFER

This macro-instruction causes the IOCS to schedule the writing of the contents of a specified area of core storage on the console printer. The IOCS then executes the next sequential instruction of the using program without waiting for execution of the scheduled write operation. The first operand of this macro-instruction is TYPE. The second operand is any label that refers to the high-order position of the area whose contents are to be written on the console printer. (A group mark with word mark must appear immediately to the right of the low-order position of this area.) The third operand is DEFER. An example of how this macro-instruction is written is shown in Figure 85.

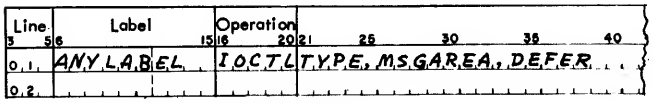


Figure 85. IOCTL TYPE, AREA, DEFER

UNCTL Macro-Instruction

There are six UNCTL macro-instructions. They may be used to generate the coding required to perform certain control functions on the input/output unit assigned to a file. These macro-instructions are discussed in the paragraphs that follow. The control functions they perform are discussed in detail in the publication, *IBM 1410 Principles of Operation*, Form A22-0526.

NOTE: The first operand of every UNCTL macro-instruction is the name of the affected file. The programmer may substitute for the name of the file, the system symbol entered as the second operand of the DTF SYMUNIT entry specified for the file. If this substitution is made, the relevant system symbol *must* be preceded and followed by a slash (e.g., /MR9/). (When this symbol is entered in the DTF SYMUNIT entry, it *may* or *may not* be preceded and followed by slashes.)

UNCTL FILE, CC, d

This macro-instruction (see Figure 86) corresponds to the CC (Control Carriage) mnemonic instruction for the 1403 Printer. The first operand is the label of the file to which the printer is assigned. The second operand is CC. The third operand is the d-character of the Control Carriage instruction.

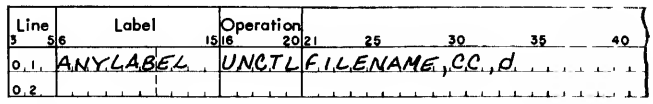


Figure 86. UNCTL FILE, CC,d

UNCTL FILE, SSF, d

This macro-instruction (see Figure 87) corresponds to the SSF (Select Stacker and Feed) mnemonic instruction for the 1402 Card Read Punch. The first operand is the name of the file to which the card reader is as-

signed; the second operand is SSF; the third operand is the d-character of the Select Stacker and Feed instruction. (If this macro-instruction is used, the DTF statement for the affected file must include the DTF ORDER entry with 9 as its operand.)

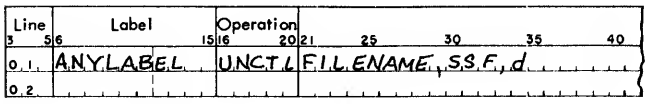


Figure 87. UNCTL FILE, SSF, d

UNCTL FILE,BSP

This macro-instruction (Figure 88) corresponds to the BSP (Backspace) mnemonic instruction for IBM magnetic tape units. The first operand is the name of the file to which the tape unit is assigned. The second operand is the Backspace mnemonic instruction.

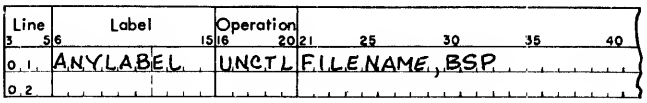


Figure 88. UNCTL FILE, BSP

UNCTL FILE, RWD

This macro-instruction (Figure 89) corresponds to the RWD (Rewind) mnemonic instruction for IBM magnetic tape units. The first operand is the name of the file to which the tape unit is assigned. The second operand is the Rewind mnemonic instruction.

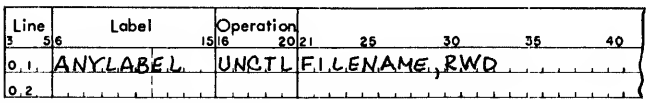


Figure 89. UNCTL FILE, RWD

UNCTL FILE,RWU

This macro-instruction (Figure 90) corresponds to the RWU (Rewind and Unload) mnemonic instruction for IBM magnetic tape units. The first operand is the name of the file to which the tape unit is assigned. The second operand is the Rewind and Unload mnemonic instruction.

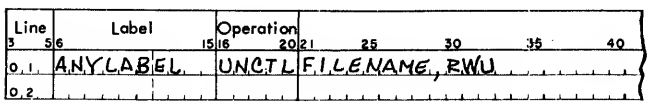


Figure 90. UNCTL FILE,RWU

UNCTL FILE,WTM

This macro-instruction (Figure 91) corresponds to the WTM (Write Tape Mark) mnemonic instruction for IBM magnetic tape units. The first operand is the name of the file to which the unit is assigned. The second operand is the Write Tape Mark mnemonic instruction.

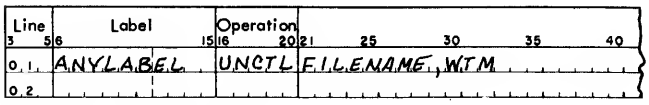


Figure 91. UNCTL FILE,WTM

Disk File Processing

This section provides the information required to use the IOCS to process disk files.

Disk Files

A disk file is a data file which the IOCS is to read from or write in 1301 Disk Storage. The first operand of the DTF SYMUNIT entry for a disk file must be 1301, and the operand of the DTF ORDER entry must specify the disk input/output instruction that is to be used to process the file.

Programs that incorporate the IOCS may only read from or write on those disk files contained in modules of 1301 Disk Storage that have been specified, at System Generation, by means of the DSKDF macro-instruction.

The SKIP operand of the DTF ERROPTNS entry may not be specified for any nonsequential disk file. The DTF EOFADDR entry must be specified for all sequential disk files. (The various sequential and nonsequential disk files are discussed in the material that follows.)

NOTE: *The operand of the DTF FILEFORM entry for nonsequential disk files must be 1 or 3.*

FORM A

A Form A (Sequential-Full Track) disk file is a disk file organized in such a way: (a) that it is to be read or written by means of the Read/Write Full Track Without Record Addresses, the Read/Write Full Track With Home Address, or the Read/Write Full Track With Addresses instruction, and (b) the IOCS is to increment the current track address by one to obtain the address of the next track to be read or written.

When the current track address of a Form A disk file has been incremented to the point where it is one higher than the address of the last track contained in the current physical unit assigned to the file, the IOCS determines: (a) whether an alternate physical unit is

to be assigned to the file, or (b) whether an end-of-file condition exists on the file. If an alternate unit is to be assigned, the IOCS links with the System Monitor to effect the assignment and then passes control to the user's program. If an end-of-file condition exists on the file, the IOCS causes a branch to be executed to the user's end-of-file routine for the file.

Input/output areas associated with Form A disk files must conform to the format shown in Figure 92.

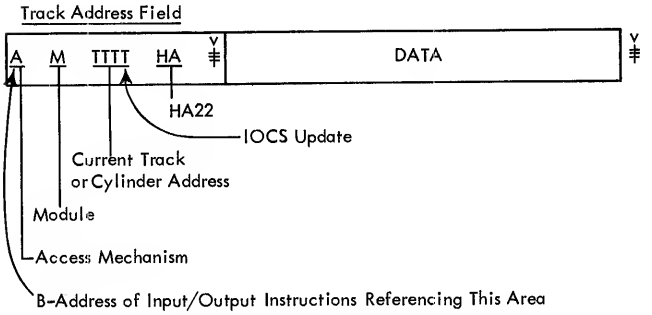


Figure 92. Input/Output Areas (Form A and B Disk Files)

The IOCS supplies the information contained in the Track Address Field shown in Figure 92. The IOCS updates the current track address at the low-order position of the subfield shown at TTTT.

FORM B

A Form B (Nonsequential-Full Track/Cylinder) disk file is a disk file organized in such a way: (a) that it is to be read or written by means of one of the Full Track instructions or by means of the Read/Write Cylinder Operation instruction, and (b) the IOCS cannot increment the current track address by one to obtain the address of the next track that is to be read or written.

NOTE: Form B disk files are the *only* disk files that may be read or written by means of the Read/Write Cylinder Operation instruction.

For Form B disk files, the IOCS makes no check to determine if an end-of-physical-unit or end-of-file condition has occurred on the file.

Input/output areas associated with Form B disk files must conform to the format shown in Figure 92.

For this type of disk file, the programmer must supply the information contained in the Track Address Field.

To read records from or write records on Form B disk files, the programmer must perform the following operations:

1. Place the address of the next disk track the IOCS is to read or write in the appropriate portion of the Track Address Field associated with the input/output area that is to handle the affected records.

2. Make sure that the B-address of the input/output instruction in the IORW at the top of the File List associated with the affected file refers to the appropriate input/output area.

3. Issue the nonsequential form of the desired GET or PUT macro-instruction. (See "Disk Macro-Instructions.") IORW's, File Lists, and File Tables are discussed under "Internal Operation of the IOCS."

Single-Record Disk Files

A single-record disk file is a disk file the IOCS is to read or write by means of the Read/Write Single Record instruction. There are five types of single-record disk files. Each of these five types is discussed in the material that follows.

FORM C

A Form C (Sequential-Geometric) disk file is a single-record disk file organized in such a way that: (a) the IOCS can increment the current record address by one to obtain the address of the next record that is to be read or written, (b) the first four digits of the address of each record of the file are identical to the first four digits of the address of the disk track on which the record resides, and (c) each track of the file contains the same number of data records.

The DTF FILEFORM entry for this type of file must include a second operand. This second operand is an integer equal to the number of data records contained on any one track of the file. An example of a FILEFORM entry that includes this operand is shown in Figure 93.

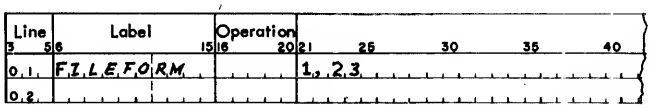


Figure 93. FILEFORM Entry (Second Operand Included)

Input/output areas associated with Form C disk files must conform to the format shown in Figure 94.

The IOCS supplies the information contained in the Track/Record Address Field shown in Figure 94.

When the number of data records read from or written on a disk track of this type of file equals the integer specified as the second operand of the DTF FILEFORM entry for the file, the IOCS: resets the subfield YY shown in Figure 94 to zeros (00), and increments the subfield xxxx by one.

If the current track address of a Form C disk file has been incremented to the point where it is one

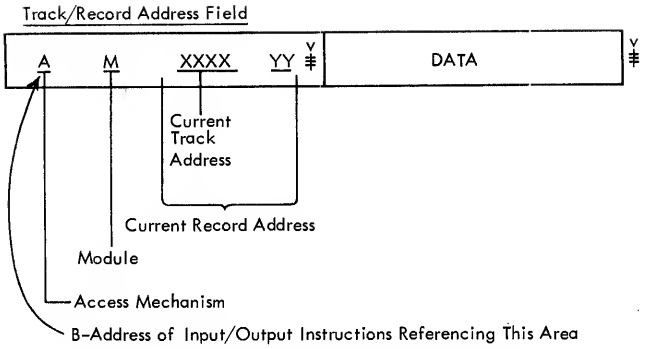


Figure 94. Input/Output Areas (Form C Disk Files)

higher than the last track of the current physical unit of the file, the IOCS performs the same functions described under "Form A."

FORM D

A Form D (Nonsequential-Geometric) disk file is a single-record disk file organized in such a way that: (a) the IOCS cannot increment the current record address by one to obtain the address of the next record to be read or written, and (b) the first four digits of the address of each record contained in the file are identical to the first four digits of the address of the disk track on which the record resides.

Input/output areas associated with Form D disk files must conform to the format shown in Figure 95.

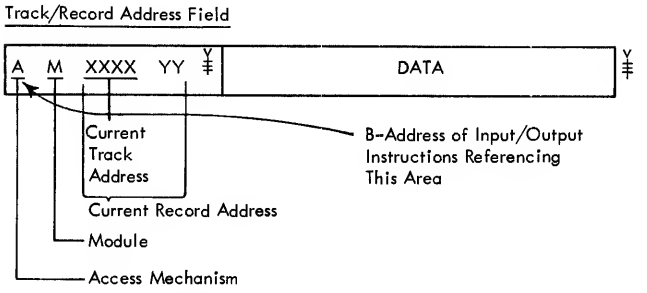


Figure 95. Input/Output Areas (Form D Disk Files)

The programmer must supply the information shown in the Track/Record Address Field illustrated in Figure 95.

In order to read records from or write records on Form D disk files, the programmer must perform the following operations:

1. Place the access mechanism, module and track address of the next logical data record in subfield AMXXXX shown in Figure 95, and place the two digits that make this track address the record address of the next logical data record in subfield YY. (The IOCS uses AMXXXX to issue the necessary Seek Disk instruction. It uses AMXXXXYY to execute the desired read or write operation.)

2. Make sure that the B-address of the input/output instruction in the top IORW on the File List associated with the affected file refers to the appropriate input/output area. (Input/Output Request Words are discussed under "Internal Use of the IOCS.")

3. Issue the nonsequential form of the desired GET or PUT macro-instruction (e. g., PUT FILE,NSEQ). See "Disk Macro-Instructions" for a discussion of nonsequential macro-instructions.

The IOCS cannot check for end-of-physical-unit or end-of-file conditions on this type of file.

FORM E

A Form E (Sequential-Nongeometric) disk file is a single-record disk file organized in such a way that: (a) the IOCS can increment the current track address by one to obtain the address of the next track from which data records are to be read or written, (b) the first four digits of the address of each record of the file are *not* identical to the first four digits of the address of the disk track on which the record resides, and (c) each track of the file contains the same number of data records.

The second operand of the DTF FILEFORM entry must be included for this type of file. This operand is discussed under "Form C."

The DTF FILEFORM entry for this type of file must also include a third operand. The required third operand is NGEOM. An example of how this entry is coded, including this third operand, is shown in Figure 96.

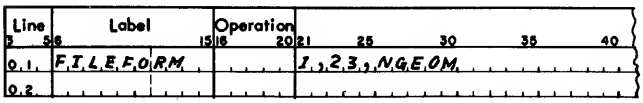


Figure 96. FILEFORM Entry (Second and Third Operands Included)

The DTF SCRAMBLE entry must be included in the DTF statement for this type of file. The operand of the SCRAMBLE entry is the label of a user-written routine (i.e., scramble routine) that provides the IOCS with the address of the next data record that is to be read from or written on the file. An example of how this entry is coded is shown in Figure 97.

When a request for a read or write operation against

a Form E disk file is noted, the IOCS performs the following operations:

1. Places the B-address of the requested input/output operation in index register 15.
2. Moves the current track address (updated if necessary) into both the affected Track Address Field and Record Address Field (Figure 98). (The IOCS follows the updating procedure outlined under "Form C" for this type of file.)
3. Causes a branch to be taken to the scramble routine provided for the file.

The user's scramble routine must then: (a) compute the address of the proper data record, (b) place this address in the relevant portion of the affected Record Address Field (see Figure 98), and (c) cause a branch to be taken back to the IOCS at /SCR/.

NOTE: The user's scramble routine can neither make use of IOCS macro-instructions nor cause the using program to enter Priority Alert. (The using program is removed from Priority Alert when the IOCS passes control to the user's scramble routine.)

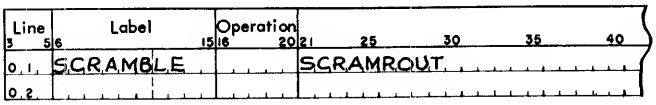


Figure 97. SCRAMBLE Entry

Input/output areas associated with Form E disk files must conform to the format shown in Figure 98.

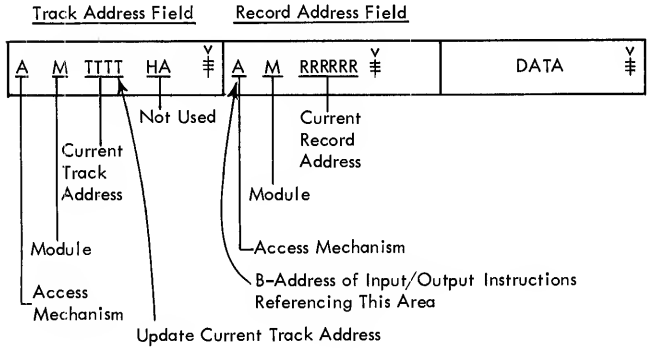


Figure 98. Input/Output Areas (Form E Disk Files)

The IOCS supplies the information contained in the Track Address Field and Record Address Field shown in Figure 98, with one exception. The programmer must provide the information contained in the current record address portion of the Record Address Field.

When the current track address of this type of file has been incremented to the point where it is one higher than the address of the last track of the physical unit currently assigned to the file, the IOCS performs the same operations it performs for Form A disk files in this situation.

FORM F

A Form F disk file (nonsequential – nongeometric) is a single-record disk file organized in such a way that: (a) the IOCS *cannot* increment the track address by one to obtain the address of the next track from which records are to be read or written, and (b) the first four digits of the address of each record of the file are *not* identical to the first four digits of the address of the disk track on which the record resides.

The DTF FILEFORM entry for this type of file must include the third operand discussed under "Form E." The second operand of the FILEFORM entry discussed under "Form C" is not applicable and must be omitted. However, the comma that would normally separate the second operand from the third operand must be included. An example of how the FILEFORM entry is coded, if the third operand is included and the second is omitted, is shown in Figure 99.

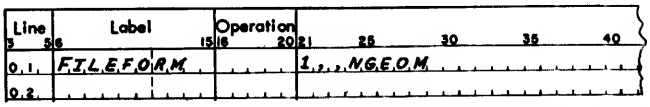


Figure 99. FILEFORM Entry (Second Operand Omitted; Third Operand Included)

Input/output areas associated with this type of file must conform to the format shown in Figure 100.

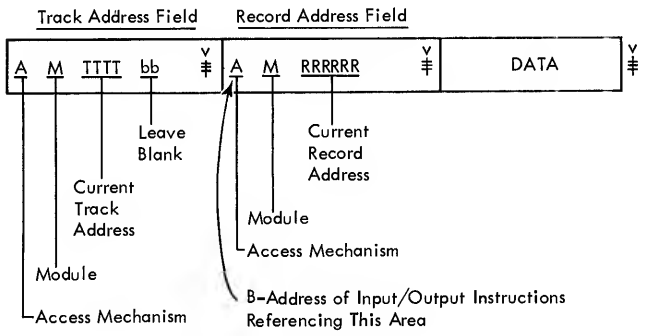


Figure 100. Input/Output Areas (Form F Disk Files)

The programmer must supply all the information contained in the Track Address Field and Record Address Field, shown in Figure 100.

To read data records from or write data records on Form F disk file, the programmer must perform the operations described under "Form D."

The IOCS makes no check to determine if an end-of-physical-unit or end-of-file condition has occurred on this type of file.

FORM G

A Form G (Partitioned Sequential-Geometric) disk file is a single-record disk file organized in such a way that: (a) the IOCS is to read or write only one data record per disk track, (b) the record that is to be read or written occupies the same relative position on each track (e.g., every record of the file is the second record on the track that contains it), and (c) the IOCS is to increment the current track address by one to obtain the address of the next record that is to be read or written, and the address of the track on which that record resides.

The programmer specifies that a file is a Form G disk file by entering 1 as the operand of the DTF ORDER entry for the file, and by omitting the second and third operands of the DTF FILEFORM entry for the file.

Input/output areas associated with this type of file must conform to the format shown in Figure 101.

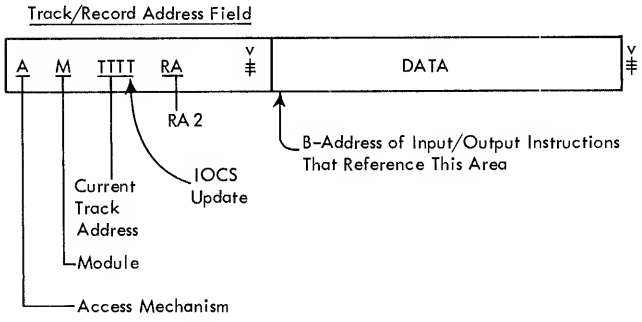


Figure 101. Input/Output Areas (Form G Disk Files)

All the information shown in the Track/Record Address Field in Figure 101 is supplied by the IOCS.

When the current track address of this type of disk file has been incremented to the point where it is one higher than the last track of the physical unit currently assigned to the file, the IOCS performs the same operations it performs in this situation for Form A disk files. See the discussion of "Form A."

Single-Record Disk File Summary: A summary of the characteristics of the various types of single-record disk files is shown in Figure 102.

NOTE: The IOCS cannot process Form C,D,E, and F disk files unless the operand 7 appears in the seventh operand field of the IOKDF macro-instruction at System Generation. No such restriction applies to Form A,B, or G disk files.

Disk Macro-Instructions

All GET and PUT macro-instructions, except PUT FILE,d; PUT WORK TO FILE,d; and PUT FILE TO FILE,d, can be issued against disk files. However, GET and PUT macro-instructions issued against Form B, D, or F disk files must include an additional operand. This operand is NSEQ. It must be separated from the preceding operand by a comma. An example of how the NSEQ operand is included is shown in Figure 103.

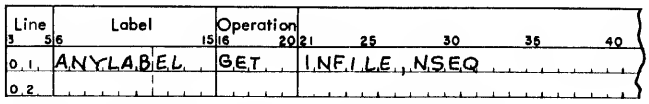


Figure 103. GET INFILE,NSEQ (Nonsequential Disk)

	IOCS Update Current Track Address	IOCS Update Current Record Address	2nd Operand FILE-FORM	3rd Operand FILE-FORM	GET and PUT NSEQ	Track Address Field	Record Address Field	Track/Record Address Field	SCRAMBLE Entry
Form C (Sequential-Geometric)	YES	YES	YES	NO	NO	NO	NO	YES	NO
Form D (Nonsequential-Geometric)	NO	NO	NO	NO	YES	NO	NO	YES	NO
Form E (Sequential-Nongeometric)	YES	NO	YES	YES	NO	YES	YES	NO	YES
Form F (Nonsequential-Nongeometric)	NO	NO	NO	YES	YES	YES	YES	NO	NO
Form G (Partitioned Sequential-Geometric)	YES	YES	NO	NO	NO	NO	NO	YES	NO

Figure 102. Single-Record Disk File Summary

This section provides descriptions of certain aspects of the internal operation of the IOCS. The following topics are discussed: Input/Output Request Word (IORW), System Monitor, File Table, IOCS Lists, and IOCS Error Procedures.

Input/Output Request Word (IORW)

The IORW is a group of contiguous fields normally generated by the IOCS. (See Figure 104.) These fields contain the information needed to define a specific input/output operation, and the status of that operation at any given point during execution of the user's program. This information is normally provided by the IOCS, by DTF entries, and by the System Monitor.

LINK FIELD

This is a five-position field that links together all the IORW's that appear on the same IOCS List. It contains the address of the low-order position of the Link Field of the next IORW on the same list. The Link Field of the last IORW on a given list contains 00000.

FILE LIST ADDRESS FIELD

This is a five-position field that associates each IORW with the File Table of a particular file. It contains the address of the low-order position of the File List Origin (i.e., Field 5 of the relevant File Table). This address may be referred to by the name of the file (i.e., the label specified in the operand of the DTF Header Line entry for the file).

INPUT/OUTPUT OPERATION FIELD

This is a ten-position field that contains a machine-language input/output instruction which the IOCS schedules and executes. The B-address of this instruction ties the IORW to a specific input/output area (i.e., the B-address of this instruction is the low-order address of the relevant input/output area).

ERROR COUNT FIELD

This is a two-position field. When an error results from execution of the instruction that appears in the Input/Output Operation Field, the IOCS uses this field to maintain a count of the number of times the instruction has been re-executed while attempting to correct the error.

S (CHANNEL STATUS CHARACTER) FIELD

This is a one-position field. The IOCS places a character in this field when execution of the instruction that appears in the Input/Output Operation Field results in an error. The BCD code of this character (i.e., the Channel Status Character) indicates the channel status indicators that were turned on by the error, with two exceptions. These execeptions are: (a) the B bit is not included in the BCD code of this character if the WLR operand of the DTF ERRCHECK entry was omitted, even if the wrong-length-record indicator is on, and (b) if the relevant input/output operation was a disk operation and no data was actually transferred by the operation; the 1 bit is included to indicate that this is the case, even though the Not Ready channel status indicator was not turned on by the operation.

D (CHANNEL TEST CHARACTER) FIELD

This is a one-position field. The character (i.e., the Channel Test Character) in this field becomes the d-modifier of a Branch External Indicator (BEX) instruction.

The IOCS uses this instruction to test for error conditions after execution of the instruction that appears in the Input/Output Operation Field.

If the WLR operand of the DTF ERRCHECK entry was specified for the file, and the file consists of Form 1 or Form 2 records, the Channel Test Character is a group mark (≡). If the WLR operand was not specified, or if the file consists of Form 3 or Form 4

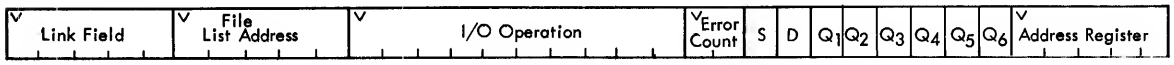


Figure 104. IORW

records, this character is a segment mark (⊣⊢). If the fourth operand of the ERRCHECK entry was specified, this character is the character that appears as that operand.

IORW INDICATORS

Fields Q1 through Q6 are one-position fields. Each of these fields contains a single character. The BCD code of these characters is used to control the internal operation of the IOCS. The bits these characters may contain, and the significance of each bit, are summarized in Figure 105.

The bits associated with each box shown in Figure 105 are mutually exclusive (e.g., if the B bit appears, the A bit may not appear). The bit associated with boxes marked SPARE is always the low-order bit shown (i.e., A, 4, or 1). The programmer must keep this fact in mind if he plans to construct IORW's himself. A similar situation applies to the box marked INTERNAL. However, once the IORW is in use, the IOCS may alter these bits.

In the paragraphs that follow, Fields Q1-Q6 are discussed in detail.

Q1: This field is reserved for the future use of the IOCS.

Q2: The BCD code of the character in this field includes the 8 bit if the second operand of the DTF FILEFORM entry was specified. If the BCD code includes the 4 bit, this operand was not specified. The BCD code of the character in this field includes the 2 bit if the NGEOM operand was specified in the FILEFORM entry for the file (i.e., the file is a Form E or Form F disk file). If the BCD code includes the 1 bit, the NGEOM operand was not specified.

Q3: The BCD code of the character in this field includes the B bit, if a user-written error routine has been provided for the file with which the IORW is associated. If the BCD code includes the A bit, no such routine has been provided. (See the DTF "ERRADDR" entry.)

The BCD code of the character in this field includes the 2 bit, if a user-written service routine has been provided for the file with which the IORW is associated. If the BCD code includes the 1 bit, no such routine has been provided. (See the DTF "INTADDR" entry.)

Q4: The BCD code of the character in this field includes the B bit, if the input/output area associated with the IORW has been released to the program (i.e., the Traffic Bit is ON). The BCD code includes the A bit, if the associated input/output area has not been released to the program (i.e., the Traffic Bit is OFF). (See "Interaction of the IOCS Lists," for a discussion of when the Traffic Bit is set ON and OFF.) The BCD code of the character in this field includes the 2 bit if the IOCS is to make a programmed wrong-length-record check after execution of the instruction in the Input/Output Operation Field. If the BCD code includes the 1 bit, this check is not made (see the DTF "ERRCHECK" entry.)

Q5: The BCD code of the character in this field includes the B bit (i.e., the Exception Bit is ON) if: (a) an error or end-of-reel condition has occurred as a

BIT CONFIGURATION	B	A	8	4	2	1
Q1	SPARE		SPARE		SPARE	
		NORMAL		NORMAL		NORMAL
Q2	INTERNAL		2-Op FILEFORM		NONGEOMETRIC	
		NORMAL	YES	NO	YES	NO
Q3	ERROR EXIT		SPARE		INTERRUPT EXIT	
	YES	NO		NORMAL	YES	NO
Q4	TRAFFIC BIT		SPARE		WLR CHECK	
	ON	OFF		NORMAL	YES	NO
Q5	EXCEPTION BIT		SPARE		STORE	
	ON	OFF		NORMAL	YES	NO
Q6	ERROR BIT		SPARE		WRITE DISK CHECK	
	ON	OFF		NORMAL	YES	NO

Figure 105. IORW Indicators

result of execution of the instruction that appears in the Input/Output Operation Field, and (b) an exit is to be taken as a result of this condition to the error or end-of-file routine provided for the file by the programmer. If the BCD code includes the A bit (i.e., the Exception Bit is OFF), no error or end-of-reel condition has occurred.

The BCD code of the character in this field includes the 2 bit if the IOCS is to store the contents of the E-address or F-address register after execution of the instruction that appears in the Input/Output Operation Field. If the BCD code includes the 1 bit, no such action is to be taken. (See the DTF "ERRCHECK" entry.)

NOTE: If the BCD code of the character in Field Q4 includes the 2 bit, the code of the character in Field Q5 also includes the 2 bit. If the BCD code of the character in Field Q5 includes the 2 bit, the BCD code of the character in Field Q4 does not necessarily include the 2 bit.

Q6: The BCD code of the character in this field includes the B bit if: (a) an error condition resulted from execution of the instruction in the Input/Output Operation Field, and (b) an exit is to be taken, as a result of this error condition, to the error routine provided for the file by the programmer. If the BCD code includes the 1 bit, no such error condition has occurred.

The BCD code of the character in this field includes the 2 bit if the IOCS is to make a Write Disk Check after execution of the instruction in the Input/Output Operation Field. If the BCD code includes the 1 bit, this check is not made. (See the DTF "ERRCHECK" entry.)

NOTE: If the BCD code of the character in Field Q5 includes the B bit, the code of the character in Field Q5 also includes the B bit. However, it should be noted that the converse is not necessarily true.

ADDRESS REGISTER FIELD

This is a five-position field. It is generated: (a) if the WLR operand of the DTF ERRCHECK entry was specified for the relevant file, and the file consists of Form 3 or Form 4 records, or (b) if the STORE operand of the DTF ERRCHECK entry was specified for the relevant file, regardless of the record form used by the file.

The IOCS stores, in this field, the length of the last record read after execution of the instruction that appears in the Input/Output Operation Field if: (a) the file consists of variable-length records, and (b) the WLR operand was specified for the file. (The IOCS compares the contents of this field to the contents of the relevant Block Character-Count to determine if a wrong-length record has been read.) In all other cases, the IOCS stores the contents of the E-address or F-address register in this field after execution of the relevant input/output operation.

User-Coded IORW's

The programmer can create an IORW by providing coding of the format shown in Figure 106.

Line	Label	Operation	21 25 30 35 40
0.1		D.C.W.	00000
0.2		D.C.W.	F.I.L.E.N.A.M.E.
0.3		M.V.	*U.O., I.O.A.R.E.A., R.
0.4		D.C.W.	@ 6 6 6 ≠ V V V V V N V @
0.5		D.C.W.	#5
0.6			

Figure 106. Coding the IORW

The operand shown on Line 1 of Figure 106 must be either 00000 or the address of the low-order position of the Link Field of another IORW. The units position of this field must always be signed plus. The operand shown on Line 2 must be the name of the file (as it appears in the operand of the relevant DTF Header Line entry) with which the IORW is to be associated. The operands shown on Line 3 must be (in order) the x-control field, the B-address (i.e., the label referring to the appropriate input/output area), and the d-modifier of the instruction that is to appear in the Input/Output Operation Field of the IORW. The operand shown on Line 4 reserves the ten positions of core storage needed to contain the Error Count, the S, the D, and the Q fields of the IORW. The first three positions (i.e., the Error Count and S fields) must contain blank characters. The last seven positions should contain the actual characters the programmer wants to appear in the D field and the Q fields. The operand shown on Line 5 is required only if the Address Register Field is to appear.

Any IORW created by the programmer can be associated with the relevant file by means of the DTF FILELIST entry.

The following restrictions apply to user-coded IORW's:

1. The Traffic Bit must be ON if the file is not to be opened before it is used (i.e., the BCD code of the character in Field Q4 includes the B bit, but not the A bit).

2. The low-order character of the x-control field of the instruction in the Input/Output Operation Field cannot be 0 or 3, if the instruction is a disk instruction.

Modification of the IORW

The contents of any field of an IORW can be modified during execution of the program, as long as the modification preserves the intent of the field.

System Monitor

The System Monitor provides the IOCS with the following information:

1. If a file uses a unit-record device, the System Monitor provides a character indicating the channel to which that device is attached (i.e., the channel character). This character is placed in the x-control field portion of the Input/Output Operation Field of each IORW on the File List associated with the file.

2. If a file uses a magnetic tape unit, the System Monitor provides: (a) the number of the unit, and (b) a character indicating the channel to which the unit is attached. This information is placed in the x-control field portion of the Input/Output Operation Field of each IORW on the File List associated with the file.

3. If a file uses a module of 1301 Disk Storage, the System Monitor provides: (a) a character indicating the channel to which the module is attached, (b) the address of the first track of the physical unit currently assigned to the file, and (c) the Ending Track Address of the physical unit currently assigned to the file. This information is placed, respectively, in: (a) the x-control field portion of the Input/Output Operation Field of each IORW on the File List associated with the file, (b) Field 3 of the File Table associated with the file, and (c) Field 4 of the File Table associated with the file.

The IOCS automatically links with System Monitor to obtain this information whenever:

1. A file is opened by means of an IOCTL OPEN macro-instruction.

2. An end-of-reel condition occurs on a tape file for which an alternate unit (or units) has been specified.

3. An end-of-physical-unit condition occurs on a disk file for which an alternate physical unit (or units) has been specified.

The information provided by the System Monitor differs as follows:

1. When a file is opened, the information provided refers to the Base Unit assigned to the file.

2. When an end-of-reel or physical-unit condition occurs on a file, the information provided refers to the next alternate unit (if any).

Monitor Calling Sequences

The System Monitor can provide the information discussed above at times other than when a file is opened, or an end-of-reel, or end-of-physical-unit condition occurs on a file. To obtain this information at times other than those noted above, the programmer must enter the appropriate System Monitor calling sequences in his object program. The various System Monitor calling sequences are discussed in the material that follows.

BASE/CURRENT SEQUENCE

The Base/Current sequence causes the System Monitor to establish the Base Unit associated with the file as the Current Unit of the file. This sequence consists of the coding shown in Figure 107.

Line	Label	Operation	
0 1		ZA	+FILENAME,,X15
0 2		BXP,A	/MIQ/
0 3			

Figure 107. Base/Current Sequence

The label FILENAME shown on Line 1 must be the same label that appears in the operand field of the DTF Header Line entry for the affected file.

ALTERNATE/CURRENT SEQUENCE

The Alternate/Current sequence causes the System Monitor to establish the next Alternate Unit associated with the file as the Current Unit of the file. This sequence consists of the coding shown in Figure 108.

Line	Label	Operation	
0 1		ZA	+FILENAME,X15
0 2		BXP,A	/MIP/
0 3		BE	ANYLABEL
0 4			

Figure 108. Alternate/Current Sequence

The label ANYLABEL shown on Line 3 of Figure 108 refers to a user-written routine that is to be exited to if there is no Alternate Unit available to be established as the Current Unit. When this exit is taken, the System Monitor automatically establishes the Base Unit as the Current Unit.

USE CURRENT UNIT SEQUENCE

The Use Current Unit sequence must be entered in the user's source program subsequent to the coding for the Base/Current sequence or the Alternate/Current sequence (whichever is appropriate). The exact

point at which the Use Current Unit sequence is entered is not critical, as long as it appears in the user's object program: (a) after the relevant Base/Current or Alternate/Current sequence, and (b) before the next GET or PUT macro-instruction that references the affected file. The Use Current Unit sequence consists of the coding shown in Figure 109.

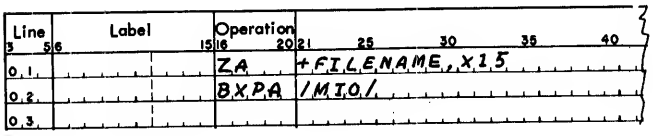


Figure 109. Use Current Unit Sequence

File Table

The File Table is a group of fields that contain information the IOCS must have to schedule and execute input/output and related operations on a file. The information contained in the File Table is provided by DTF entries and the System Monitor. When a DTF statement is not provided for a file, the programmer must construct the File Table for that file.

The fields of the File Table (Figure 110) are discussed in the section that follows.

Tape/Disk Prefix

The File Table Tape/Disk Prefix consists of Fields 1-4. If the file is a *tape* file all four fields appear. If

Field	Mnemonic	Operand	Explanation
	(Disk File Prefix)		
1	DCW	xxxxx	Address af user's Scramble Routine
2	DCW	xx	Number records per track
	DCW	xx	Current Recard
3	DCW	AMTTTTHA	Starting Address
4	DCW	TTTT	Ending address
	(All Devices)		
5 FILENAME	DCW	xxxxx	File List Origin
6	DCW	/xxx/	Symbalic Unit
7	DCW	xxxxx	User EOF rautine address
8	DC	@x@	File Indicatar 1
	DC	@x@	File Indicatar 2
	DC	@x@	File Indicatar 3
	DC	@x@	File Indicatar 4
	DC	@x@	File Indicatar 5
	DC	@x@	File Indicatar 6
9	DCW	xx	Index register address
10	DCW	xxxxx	Miscellaneous. 1
11	DCW	xxxxx	Miscellaneaus 2
12 LENGTHLBL	DCW	xxxxx	Miscellaneaus 3
13	DCW	xxxx	Miscellaneaus 4
14	DC(W)	@x@	Errar Indicatar 1
	DC(W)	@x@	Errar Indicator 2
	DC(W)	@x@	Errar Indicator 3
	DC(W)	@x@	Errar Indicator 4
	DC(W)	@x@	Error Indicatar 5
15	DCW	xxxxx	User error rautine address
16	DCW	xxxxx	User interrupt routine address
	(File Table Extensian)		
17	DCW	@x@	Terminate table lookup character
18	DCW	xxxxx	Label routine address
	DC	@xx@	Label exit indicatar
	--		
	--		
	--		
	DCW	xxxxx	Label rautine address
	DC	@xx@	Label exit indicatar
19 EXTENLBL	DC	@x@	Label Indicator 1
	DC	@x@	Label Indicatar 2
	DC	@x@	Label Indicator 3
	DC	@x@	Label Indicator 4
	DC	@x@	Label Indicator 5
20	DCW	@x@	Record Farm
21	DCW	@xxxx@	Retention Period
22	DCW	@xxxxx@	Creatian date
23	DCW	@xxxxxxxxxx@	File identifier
24	DCW	@xxxxx@	File serial number
25	DCW	@xxxx@	Reel sequence number

Mnemonic	Operand	Explanatian
(Tape File Prefix)		
DCW	EXTENLBL	Address of File Table Extension
DCW	xxxxx	Block caunter.
DCW	xx	Reel caunter.
DCW	@RNUR@	RWD/U aptions.

Figure 110. Fields of the File Table

the file is a *disk* file, and the second and third operands of the DTF FILEFORM entry have been specified for the file, all four fields appear. If the file is a disk file, and the second, but not the third, operand of the DTF FILEFORM entry has been specified for the file, both subfields of Field 2 appear, as do Fields 3 and 4. For all other disk files, only the second subfield of Field 2, and Fields 3 and 4 appear.

FIELD 1

For Form E *disk* files, this is a five-position field that contains the address of the programmer's scramble routine. For all other types of *disk* files, this field is omitted. For *tape* files that use labels the IOCS is to process, this is a five-position field that contains the address of the File Table Extension that is to be used to process the labels of the file. For all other tape files this is a one-position field that contains a zero.

FIELD 2

For Form C and E *disk* files, this is a four-position field. In this case the first two positions of this field are a subfield that contains the number specified as the second operand of the DTF FILEFORM entry; the last two positions are a subfield that initially contains zeros. Every time a disk record is read or written, the integer 1 is added to the second subfield. When the number in the second subfield equals the number in the first subfield, the second subfield is reset to contain zeros, and the Current Track Address (Field 3) is incremented by 1. If the file is a Form A, B, D, F, or G disk file, this is a two-position field that is reserved for the internal use of the IOCS. For *tape* files, this is a five-position field that contains a count of the records (unblocked files) or blocks of records (blocked files) read from or written on the current reel of tape. Noise records (i.e., records less than 13 characters in length), header and trailer labels, and tape marks are not included in this count.

FIELD 3

For *disk* files, this is an eight-position field that contains the information shown in Figure 111 (i.e., the Current Track Address of the current physical unit). This information is initially supplied by the System Monitor.

For input *tape* files, this is a two-position field that initially contains zeros. The programmer may place a signed or unsigned positive integer in this field that represents the number of reels that make up the file. Every time the IOCS reads a complete reel of the file it decrements this integer by one. When this field contains all zeros, the IOCS causes a branch to be taken to the programmer's end-of-file routine. If this field has been set to a positive integer by the programmer, the IOCS ignores any LEOFb or LEORb label identifier encountered in the trailer label of a reel of the file. If the file is an output tape file this field is not referenced by the IOCS.

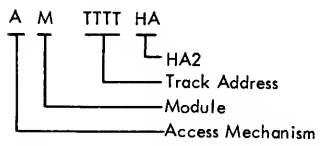


Figure 111. Current Track Address

FIELD 4

This is a four-position field. For *disk* files, this field contains the address (i.e., the End Track Address) of the last track of the current physical unit that is to be read or written by the IOCS. This address is provided by the System Monitor. It must be in the same module as the first track of the current physical unit. The IOCS uses this field to determine when the last track of the current physical unit has been read or written.

For *tape* files, this field contains the characters specified in the four operand fields of the DTF RWDOPTNS entry.

File Table Body

The File Table Body consists of Fields 5-13. These fields appear regardless of the type of input/output device used by the file.

FIELD 5

This is a five-position field that is referred to as the File List Origin. It contains the address of the first IORW on the File List associated with the file. (See the discussions of List Origin and File List, under "The IOCS Lists.") The units position of this field may be referred to by the label entered as the operand of the DTF Header Line entry for the file. When there are no IORW's on the File List, this field contains 00000.

FIELD 6

This is a five-position field that contains the address equivalent of the symbolic unit name entered as the second operand of the DTF SYMUNIT entry. This address is used by the System Monitor to provide certain information that is inserted in the IORW's associated with the file. (See the section, "System Monitor.") If the file is a disk file the System Monitor also uses this field to provide the information contained in Fields 3 and 4.

FIELD 7

This is a five-position field that contains the address of the programmer's end-of-file routine. If the DTF EOFADDR entry is omitted, this field contains zeros.

FIELD 8

This is a six-position field that contains the File Indicators F1-F6. Each indicator is a single character (see Figure 112). The bits associated with the boxes marked SPARE in Figure 112 are always the low-order bits shown (i.e., A, 4, or 1). The bits associated with each box are mutually exclusive (e.g., if the 4 bit appears the 8 bit may not appear). The file indicators are discussed in the paragraphs that follow.

F1: This indicator is the character specified by the DTF PADCHAR entry written for the file. It is a blank character (b) if the PADCHAR entry is omitted.

F2: If the BCD code of this indicator includes the B bit, the Error Accept bit is said to be ON. If the BCD code includes the A bit, the Error Accept bit is said to be OFF. The IOCS sets the Error Accept bit ON just before exiting to the error routine the user has provided for the file. If the programmer *does not* set the Error Accept bit OFF before exiting from this error routine, the IOCS, immediately after the error routine has been exited from, accepts the results of the input/output operation, the execution of which caused the error routine to be entered. The IOCS does this by setting the Traffic Bit of the affected IORW (i.e., the IORW that is then at the top of the File List associated with the file) OFF and placing the IORW on the bottom of the File List of the affected file. If the programmer *does* set the Error Accept bit OFF before exiting from his error routine, the IOCS, immediately after the error routine has been exited from, re-executes the input/output operation, the execution of which caused the error routine to be entered.

F3: When this indicator is assembled, its BCD code includes the A bit. If the user replaces this A bit with a B bit, closes and then reopens the file, the System Monitor assigns to the file the last physical unit referenced by the file. (Normally, the System Monitor assigns the Base Unit to a file when the file is opened.)

The BCD code of this indicator includes the 1 bit when it is assembled. If the user replaces this bit with a 2 bit and an end-of-reel or physical-unit condition is forced or encountered on the file, the next alternate physical unit is not assigned to the file by the System Monitor, as would normally be the case. Instead, the System Monitor assigns to the file the last physical unit referenced by the file prior to the end-of-reel or physical-unit condition. In addition, if the file is a disk file the IOCS will cause a branch to be taken to the user's end-of-file routine for the file.

F4: If the BCD code of this indicator includes the B bit, the SKIP operand of the DTF ERROPTNS entry has been specified for the file. If the BCD code includes the A bit, the ACCEPT operand of the DTF ERROPTNS entry has been specified for the file, or the ERROPTNS entry has been omitted altogether. If the BCD code of this indicator includes the 2 bit, the DTF FILEFORM entry for the file specified that the file consists of Form 4 data records. If the BCD code includes the 1 bit, the DTF FILEFORM entry for the file specified that the file consists of Form 1, 2, or 3 data records.

F5: If the BCD code of this indicator includes the B bit, an error or end-of-file condition has occurred on the file as a result of the last input/output operation performed on the file. If the BCD code includes the A bit, no such condition occurred as a result of the last input/output operation. If the BCD code of this indicator includes the 2 bit, the DTF FILEFORM entry specified that the file consists of blocked data records (i.e., Form 2 or Form 4). If the BCD code includes the 1 bit, the DTF FILEFORM entry specified that the file consists of unblocked records (i.e., Form 1 or Form 3).

F6: This indicator notifies the IOCS what type of input/output device is to be used by the file. This information is obtained from the DTF SYMUNIT entry.

The possible device indicators are:

INDICATOR	EXPLANATION
1	1402 Card Reader, 1442 Serial Card Reader, 1011 Paper Tape Reader
2	1402 Card Punch, 1403 Printer
4	7330, 729 Magnetic Tape Units
8	1301 Disk Storage

FIELD 9

This is a two-position field. It contains the last two digits of the address of the index register specified in the DTF INDEX entry. If the DTF INDEX entry is omitted, this field contains zeros.

FIELD 10

This is a five-position field that initially contains zeros. If the DTF INDEX entry was specified, this field contains the high-order address of the input/output area currently available to the user's object program. If the DTF INDEX entry was not specified, this field contains: (a) the high-order address of the data record currently available to the using program, if the file is an input file, or (b) the high-order address of that portion of the current output area that is available to the using program, if the file is an output file. The IOCS provides this information during execution of GET, PUT, IOCTL OPEN, IOCTL RELSE, and IOCTL FEOR.

If an exit has been taken to a user-written error or end-of-file routine, this field contains the return address to the series of instructions generated by the GET or PUT macro-instruction that caused the exit to be taken.

BIT CONFIGURATION	B	A	8	4	2	1
F1	PADDING CHARACTER					
F2	ERROR ACCEPT		SPARE		SPARE	
	ON	OFF		NORMAL		NORMAL
F3	RESET TO BASE		SPARE		SET TO ALTERNATE	
	NO	YES		NORMAL	NO	YES
F4	SKIP		SPARE		FORM 4	
	YES	NO		NORMAL	YES	NO
F5	EXCEPTION BIT		SPARE		BLOCKED	
	ON	OFF		NORMAL	YES	NO
F6	DEVICE CHARACTER					

Figure 112. File Indicators

FIELD 11

This is a five-position field that contains the information entered in the operand of the DTF BLOCKSIZE entry. If the file consists of Form 1 or Form 3 records, this field is not used by the IOCS. In these instances, the programmer is free to make use of this field for his own purposes.

FIELD 12

This is a five-position field. If the file is an input file made up of Form 2 or Form 3 records, it contains the low-order address +1 of the block of data records currently available to the using program. (The IOCS uses this information during execution of GET, PUT, IOCTL OPEN, IOCTL CLOSE, and IOCTL FEOR.) If the file is an output file: (a) that is made up of Form 4 records, (b) for which the DTF LENGTH entry has been specified, and (c) that is to be processed by the PUT FILE,FORM4 macro-instruction, the programmer places in this field the length of the next record he is going to move into the current output area. (Under these conditions this field may be referred to symbolically by the operand of the DTF LENGTH entry.) The two high-order positions of this field are also used by the IOCS as temporary storage for the last two characters of the current GET/PUT calling sequence (i.e., the coding generated by the current GET or PUT).

FIELD 13

This is a four-position field. If the file consists of Form 2 records, this field contains the information entered in the operand of the DTF RECSIZE entry. If the file is an output file that consists of Form 4 records, this field contains an integer equal to the number of positions in the current output area of the file that are occupied by data.

This field is not used by the IOCS if the file consists of Form 1 or Form 3 records, or if the file is an input file that consists of Form 4 records. In these instances, the programmer is free to make use of the field for his own purposes.

FIELD 14

This is a five-position field that contains the information entered in the second through sixth operands of the DTF ERRADDR entry. (The mnemonic used by the IOCS to generate a character for this field is DCW, if the corresponding operand calls for an exit bypassing standard IOCS error procedures. It is DC, if the corresponding operand calls for an exit only when the error condition or conditions specified cannot be corrected by the standard IOCS error procedures.) The IOCS uses this information to determine under what conditions it is to branch to the error routine provided by the user for the file. This field must contain blanks if the DTF ERRADDR entry is omitted.

FIELD 15

This is a five-position field that contains the address of the error routine the programmer has provided for the file. (This field can be used by the programmer for his own purposes if the DTF ERRADDR entry is omitted and the DTF INTADDR entry is included. If both DTF ERRADDR and INTADDR entries are omitted, this field does not appear.)

FIELD 16

This is a five-position field that contains the address of the service routine the programmer has provided for the file. This field does not appear if the DTF INTADDR entry is omitted.

File Table Extension

The File Table Extension consists of Fields 17-25. The IOCS uses the information contained in these fields to process the tape labels used by the file. If the file does not use labels, these fields do not appear. Most of the information contained in these fields is provided by the DTF LABEL and CHECK entries for the file (e.g., the LABEL entry supplies the information for Fields 21-25). The programmer may update or alter the information contained in these fields (see "Appendix E"). When these fields are generated, they precede the balance of the File Table in core storage, but they do not necessarily *immediately* precede the balance of the File Table.

FIELD 17

This is a one-position field that contains a character normally supplied by the IOCS. This character is used to terminate the table lookup operations the IOCS performs on Field 18. If the programmer has specified the DTF LINE1 ABOVE entry for the file, that entry supplies this character.

FIELD 18

This field contains from one to eight entries if the DTF LINE1 or LINE2 entries have been specified; from 9 to 16 entries if LINE1 ABOVE was specified. Each entry consists of a DCW followed by a DC. The DCW specifies the address of a label routine written by the user. The DC specifies the point in the IOCS tape label routines at which the IOCS is to branch to that routine.

FIELD 19

This is a five-position field that contains the Label Indicators L1-L5. Each of these indicators consists of a single character. (See Figure 113.) The bits associated with the boxes marked SPARE in the figure are always the low-order bits shown (i.e., A, 4, or 1). The bits associated with each box are mutually exclusive (e.g., if the 2 bit appears, the 1 bit may not appear).

The label indicators are discussed in the material that follows.

L1: The BCD code of this indicator includes the 1 bit if the file serial number has been entered as an operand of the DTF LABEL entry. The BCD code includes the 2 bit if the file serial number has not been entered. (If the BCD code includes the 2 bit and the file is a tape output file, the IOCS places the reel serial number of the first reel of the file in Field 24 of the File Table prior to writing the new output header label. If the BCD code includes the 1 bit, the IOCS places the specified file serial number in Field 24.)

L2: The BCD code of this indicator includes the B bit if NONSTD appears as the second operand of the DTF LABEL entry for the file. If the file uses standard labels, the BCD code of this indicator includes the A bit. The BCD code of this indicator includes the 2 bit if 120 appears as the first operand of the DTF LABEL entry for the file. The BCD code includes the 1 bit if 80 appears as the first operand, or the DTF LABEL entry was not specified for the file.

BIT CONFIGURATION	B	A	8	4	2	1
L1	SPARE		SPARE		SUBSTITUTE SERIAL	
		NORMAL		NORMAL	YES	NO
L2	NONSTD		SPARE		120-CHARACTER LABELS	
	YES	NO		NORMAL	YES	NO
L3	CHECK FILE SERIAL		SPARE		CHECK FILE IDENTIFIER	
	YES	NO		NORMAL	YES	NO
L4	CHECK REEL SEQUENCE		SPARE		CHECK CREATION DATE	
	YES	NO		NORMAL	YES	NO
L5	CHECK BLOCK COUNT		SPARE		CK RETENTION CYCLE	
	YES	NO		NORMAL	YES	NO

Figure 113. Label Indicators

L3: The BCD code of this indicator includes the B bit if the SER operand is specified in the DTF CHECK entry for the file. The BCD code includes the A bit if this operand is not specified, or the DTF CHECK entry was not specified for the file. The BCD code of this indicator includes the 2 bit if the ID operand is specified in the DTF CHECK entry for the file. The BCD code includes the 1 bit if this operand is not specified or the CHECK entry was omitted.

L4: The BCD code of this indicator includes the B bit if the SEQ operand is specified in the DTF CHECK entry for the file. The BCD code includes the A bit if this operand is not specified, or the CHECK entry was omitted.

The BCD code of this indicator includes the 2 bit if the DAT operand is specified in the DTF CHECK entry for the file. The BCD code includes the 1 bit if this operand is not specified, or the CHECK entry was omitted.

L5: The BCD code of this indicator includes the B bit if the CNT operand is specified in the DTF CHECK entry for the file. The BCD code includes the A bit if this operand is not specified, or the CHECK entry was omitted. The BCD code of this indicator includes the 2 bit if the RET operand is specified in the DTF CHECK entry for the file. The BCD code includes the 1 bit if this operand is not specified, or the CHECK entry was omitted.

ALL: If the ALL operand is included in the DTF CHECK entry for the file, the BCD code of indicators L3, L4, and L5 is B-4-2.

FIELD 20

This is a one-position field that contains a character indicating the form of the records on the file. F indicates Form 1, W indicates Form 2, B indicates Form 3, and X indicates Form 4.

FIELD 21

This is a four-position field that contains the retention period. If the file uses 80-character labels the high-order position of this field contains a –

FIELD 22

This is a five-position field that contains the creation date.

FIELD 23

This is a ten-position field that contains the file identification.

FIELD 24

This is a five-position field that contains the file serial number.

FIELD 25

This is a four-position field that contains the reel sequence number. If the file uses 80-character labels the high-order position of this field contains a –.

IOCS Lists

An IOCS List is a collection of IORW's grouped by the IOCS for a specific purpose (e.g., to schedule assignment of access mechanisms to IORW's that represent requests for disk input/output operations).

List Origin: Each IOCS list has a List Origin. This is a five-position field that contains the address of the low-order position of the Link Field of the first IORW on that list. If the list is temporarily without IORW's the List Origin contains 00000.

The Link Field of each IORW on an IOCS list contains the address of the low-order position of the Link Field of the next IORW on the same list. There is one exception to this rule. The Link Field of the last IORW on a list contains 00000 or 00000. These zeros indicate to the IOCS that the end of the list has been reached.

In addition to a List Origin, every Read/Write List maintains a List End Address. This is a five-position field that contains the address of the low-order position of the Link Field of the last IORW on the list. If a Read/Write List is temporarily without IORW's, the List End Address contains the address of the low-order position of the List Origin of that list.

FILE LISTS

One File List is associated with every File Table. (This is a function of the DTF IOAREAS or the DTF FILELIST entry.) The IORW's on File Lists associated with sequential files represent completed input/output operations. The input/output area associated with the first IORW on such a File List is the input/output area currently available to the user's program. The IORW's on File Lists are processed in the order in which they appear on these lists. The List Origins of File Lists are automatically generated in the File Tables with which the lists are associated.

DISK MODULE LISTS

There is one Disk Module List for each module of 1301 Disk Storage defined at System Generation. The List Origins of these lists are contained in the Resident IOCS. The IORW's on these lists: (a) represent requests for Seek Disk operations, or (b) indicate that the relevant Seek Disk operation, or its associated input/output operation, is in the process of being executed by the IOCS.

The last IORW on such a list contains 00000 in its

Link Field. (This is the only instance when 00000 may appear in the Link Field of an IORW.) If the list is void, the List Origin of the list contains 00000.

READ/WRITE LISTS

The IOCS maintains one Read/Write List for each data channel available to using programs. The IORW's on Read/Write Lists represent requests for input/output operations. They are normally processed in the order in which they appear on these lists. This is not the case, if: (a) a GET or PUT other than a GET,DEFER or PUT,DEFER is issued and (b) the IORW associated with the requested input/output operation is not available on the relevant File List. Under these conditions, the IOCS causes the appropriate Read/Write List to be searched until the affected IORW is located. If the affected IORW is not at the top of the list, it is immediately moved to the top. This ensures that the requested operation is the next operation executed on the affected channel. The List Origins of Read/Write Lists are contained in the Resident IOCS.

SERVICE LISTS

A Service List is associated with every service routine whose label appears as the operand of a DTF INTADDR entry. The IORW's on these lists represent requests pending for entry into user-written service routines.

The List Origins of these lists immediately precede the service routines with which they are associated. They may be referred to by the label of the relevant service routines.

PROGRAM LIST

The IOCS maintains one Program List. The entries on this list are not IORW's. They are five-position fields that contain addresses which identify service routines that have IORW's on their associated Service Lists. The Resident IOCS places these entries on the Program List, the most recent entry at the top of the list. This is a departure from the procedure followed in adding entries to the other IOCS Lists. When a new entry is added to the other IOCS Lists, it is added to the bottom of the list.

When all the IORW's on the Service List of the routine associated with the entry at the top of the Program List have been processed, the Resident IOCS removes the affected entry from the Program List. It then places the next entry at the top of the list and causes the service routine associated with that entry to be entered. This process is repeated until the Program List is devoid of entries. When the list becomes void, the Resident IOCS restores the status of the user's main-line program and causes a branch to be taken to the interrupted instruction in the user's main-line program. The List Origin of the Program List is in the Resident IOCS.

Interaction of the IOCS Lists

The flow of IORW's between the IOCS Lists is discussed in the paragraphs that follow.

When the IOCS encounters a GET or PUT macro-instruction in the user's program, the Resident IOCS scans the appropriate File List until it encounters an IORW whose Traffic Bit is OFF. The Resident IOCS then sets the Traffic Bit of the affected IORW ON and makes the input/output area associated with the IORW available to the user's program.

When the Resident IOCS scans the File List, it sends all the IORW's whose Traffic Bits are ON to the bottom of the appropriate Read/Write List. (See Path 1, Figure 114.)

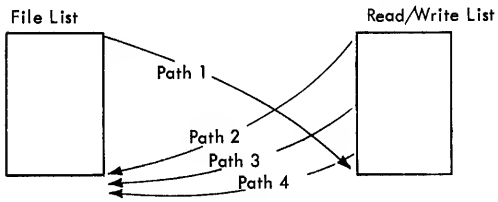


Figure 114. Interaction Between File Lists and Read/Write Lists

NOTE: In the case of disk files, when the Resident IOCS scans a File List, it sends all the IORW's whose Traffic Bits are ON to the proper Disk Module List, rather than to the appropriate Read/Write List. (See Path 1, Figure 115.)

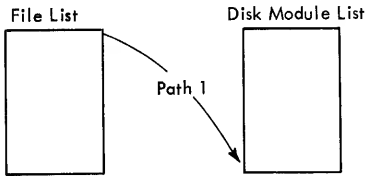


Figure 115. Interaction of File Lists and Disk Module Lists

When an IORW that has been sent to a Read/Write List reaches the top of that list (i.e., the IORW's that preceded it have been processed and removed from the list), execution of the instruction in the Input/Output Operation Field of the IORW is begun. The IORW is then removed from the Read/Write List and saved until execution of the relevant instruction has been completed. If execution of the instruction does not result in an error, the Resident IOCS sets the Traffic Bit of the affected IORW OFF, and places it on

the bottom of the relevant File List. (See Path 2, Figure 114.)

If execution of the instruction resulted in an error condition, or combination of conditions, that the programmer has specified are to cause the IOCS to: (a) bypass the normal IOCS error correction procedures, and (b) exit to the error routine the programmer has provided for the file, the Resident IOCS performs the following functions:

1. Returns the affected IORW to the bottom of the appropriate File List. (See Path 2, Figure 114.)
2. Removes all other IORW's associated with the affected file from the relevant Read/Write List (starting at the top of the list), and places them (in the order in which they are removed) on the bottom of the File List. (See Paths 3 and 4, Figure 114.)

If execution of the instruction resulted in an error condition, or combination of conditions, that the programmer has specified are to cause the IOCS to: (a) execute the normal IOCS error procedures, and (b) enter the error routine the programmer has provided for the file only if the error or errors could not be corrected by the normal error procedures, the Resident IOCS:

1. Performs the functions described in item 2 above, if the error condition or conditions could not be corrected, or
2. Places the affected IORW on the bottom of the appropriate File List (see Path 2, Figure 114), if the error condition or conditions were corrected.

If execution of the instruction resulted in an error condition, or conditions, for which the programmer has not specified an exit to the error routine he has provided for the file, the affected IORW is placed on the bottom of the appropriate File List (see Path 2, Figure 114). This is not the case, however, if the SKIP operand of the DTF ERROPTNS entry was specified for the file. If the SKIP operand was specified, the affected IORW is retained at the top of the appropriate Read/Write List so that the relevant instruction can be re-executed by the IOCS.

If execution of the instruction: (a) did not result in an error condition, or combination of conditions, that the programmer has specified are to cause the IOCS to exit to the error routine the programmer has provided for the file, (b) did not result in an error condition of any kind if the SKIP operand of the DTF ERROPTNS entry has been specified for the file, and (c) the programmer has specified that a user-written service routine has been provided for the file, the Resident IOCS takes the affected IORW off the top of the Read/Write List (see Path 1, Figure 116) and places it on the bottom of the appropriate Service List. The Resident IOCS then causes the service routine provided for the file to be entered. At this time, the Resident IOCS sets the Traffic Bit of the IORW OFF.

The programmer may perform any function he desires within a service routine; however, the last instruction of every user-written service routine must be the IOCTL EXIT,ACCEPT or the IOCTL EXIT,DELETE macro-instruction. If the IOCTL EXIT,ACCEPT is the last instruction, the Resident IOCS takes the affected IORW off the top of the relevant service routine Service List (see Path 2, Figure 116) and places it on the bottom of the appropriate File List. If the last instruction is IOCTL EXIT,DELETE the Resident IOCS notes: (a) that the IORW has been removed from the relevant Service List (see Path 3, Figure 116), and (b) that control of the IORW has been assumed by the user.

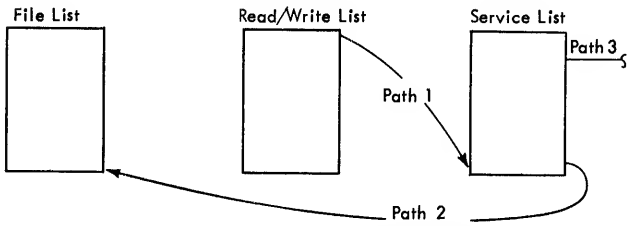


Figure 116. Interaction of File Lists, Read/Write Lists and Service Lists

IOCS Error Procedures

Upon completion of every input/output operation performed on a file, a BEX (Branch External Indicator) instruction is executed. The character found in the D (Channel Test Character) Field of the relevant IORW becomes the d-character of this instruction.

If a branch is taken as a result of executing the BEX instruction (i.e., a channel status indicator, whose associated BCD bit matches a bit included in the BCD code of the d-character of this instruction, was turned on as a result of the input/output operation), the IOCS Error Control routine is entered. If a branch is not taken, the next sequential instruction of the user's program is executed.

Error Control Routine

The primary function of the Error Control routine is to determine if and when a particular IOCS error routine is to be entered. The operation of the Error Control routine, prior to entry into a particular IOCS error routine, is:

1. When control passes to the routine, it stores the Channel Status Character (i.e., a character whose BCD code indicates which channel status indicators

were turned on upon completion of the relevant input/output operation) with appropriate alterations, in the S Field of the affected IORW. Alterations to the Channel Status Character are discussed under "S (Channel Status Character) Field."

2. The Error Control routine then attempts to match the Channel Status Character with one of the characters in Field 14 of the relevant File Table that is tagged with a word mark.

3. If the Channel Status Character does not match a character in Field 14 that is tagged with a word mark, the Error Control routine causes a branch to be taken to the appropriate IOCS error routine.

4. If the Channel Status Character matches a character in Field 14 that is tagged with a word mark, the Error Control routine examines indicator Q3 of the affected IORW to determine if a user-written error routine has been provided for the relevant file.

5. If no such routine has been provided, the IOCS error routines are bypassed and the affected input/output operation is processed as if it were error free.

6. If a user-written error routine has been provided, the Error Control routine: (a) sets the Exception Bit and Error Bit in the affected IORW ON, (b) places the affected IORW on the bottom of the relevant File List with its Traffic Bit set OFF, and (c) removes all the IORW's associated with the affected file from the appropriate Read/Write List (starting at the top of the list) and places them (in the order in which they were removed from the Read/Write List) on the relevant File List (Traffic Bits set ON).

The Error Control routine also performs certain operations if an IOCS error routine is unable to correct an error condition directed to it for correction. These operations are:

1. When an IOCS error routine cannot correct an error, control passes to the Error Control routine which attempts to match the Channel Status Character in the affected IORW with a character in Field 14 of the relevant File Table.

2. If the Channel Status Character in the S Field of the IORW does not match a character in Field 14, the Error Control routine checks the fourth File Indicator (F4) of the relevant File Table.

3. If the BCD code of F4 includes the B bit (i.e., the SKIP operand of the DTF ERROPTNS entry has been specified for the file), the Error Control routine retains the affected IORW at the top of the relevant Read/Write List with its Traffic Bit set ON (i.e., schedules immediate re-execution of the instruction that appears in the Input/Output Operation Field of the IORW).

4. If the BCD code of F4 includes the A bit (i.e., the ACCEPT operand of the DTF ERROPTNS entry has been specified for the file, or the ERROPTNS entry has been omitted altogether), the Error Control routine places the affected IORW on the bottom of the relevant File List with its Traffic Bit set OFF.

5. If the Channel Status Character matches a character in Field 14, the Error Control routine examines Field Q3 of the affected IORW to determine if a user-written error routine has been provided for the file.

6. If such a routine has been provided, the Error Control routine: (a) sets the Exception Bit and the Error Bit in the affected IORW ON, (b) places the affected IORW on the bottom of the appropriate File List with its Traffic Bit set OFF, and (c) removes all the IORW's associated with the affected file from the appropriate Read/Write List (starting at the top of the list) and places them (in the order in which they were removed from the Read/Write List) on the relevant File List (Traffic Bits set ON).

7. If this routine has not been provided, the affected input/output operation is processed as if it were error free.

Error Routines

The IOCS provides an error routine for each type of input/output device specified by the using installation at System Generation. Each IOCS error routine attempts to correct the error conditions routed to it for correction by re-executing the relevant input/output operations, or by other means peculiar to the affected type of input/output device.

Whenever an IOCS error routine causes an input/output operation to be re-executed, the number in the Error Count Field of the affected IORW is incremented by one and the Channel Status Character of the IORW is replaced with a blank character. If the error recurs, the operation is retried until the number in the Error Count Field equals a maximum value that is determined by the type of device and whether the device is involved in reading or writing data. When this maximum value is reached, the error is considered uncorrectable and the affected error routine causes a branch to be taken to the Error Control routine.

If the error is corrected, the affected IOCS error routine places the relevant IORW on the bottom of the appropriate File List with its Traffic Bit set OFF.

Appendixes

Appendix A: Shared Input/Output Areas

Two programming schemes that allow an input and an output file to share the same input/output areas are discussed in this section.

Two File/Three Area Overlap Programming

A programming scheme that allows one input and one output file sharing the same three input/output areas to be processed in a manner that provides for the overlapping of read/write and processing operations is discussed in the paragraphs that follow. This scheme allows records to be read into, processed in, and written out of the same input/output area.

To use this scheme, the program must be initialized as follows: (All labels used in this example are for illustrative purposes only.)

1. Two files, one an input file labeled INFILE, one an output file labeled OUTFILE, are defined.

2. Three input/output areas, labeled AREA1, AREA2, and AREA3, respectively, are defined.

3. Three IORW's are generated, and one is labeled. Two of the IORW's are generated by means of the DTF IOAREAS entry included in the DTF statement defining INFILE, with AREA1 and AREA2 as its first and second operands. The third IORW is generated and labeled by means of the DTF IOAREAS entry included in the statement defining OUTFILE, with AREA3 as its first operand and OUTIORW as its fourth operand.

4. INFILE and OUTFILE are opened (see Lines 1 and 2 of Figure 117).

5. The Traffic Bit of the IORW labeled OUTIORWA is set OFF (see Line 3).

Once the program has been initialized, the two files are processed as follows:

1. The IOCS is requested to make the next logical record from INFILE available in the specified input/output area (see Line 4).

2. The last record made available by the IOCS is processed (see Line 5).

3. The program is taken out of Priority Alert (see Line 6).

4. The address of the low-order position of the Link Field of the IORW on the top of the File List associated with INFILE is stored in index register 15 (see Line 7).

5. The IORW on the top of the File List associated with INFILE is removed from that list and the next IORW on the list (if any) is moved to the top of the list (see Line 8).

6. The address of the File List Origin associated with OUTFILE is placed in the File List Address Field of the IORW that was removed from the File List associated with INFILE by Step 5 (see Line 9).

7. The IORW that was at the top of the File List associated with OUTFILE (if any) is linked to the IORW that is to be placed on the File List associated with OUTFILE by Step 8 (see Line 10).

8. The IORW that was removed from the File List associated with INFILE by Step 5 is placed at the top of the File List associated with OUTFILE (see Line 11).

9. The x-control fields of the IORW's linked to the File List of OUTFILE are initialized by the System Monitor (see Lines 12 and 13).

10. The IOCS is requested to write the last record processed by the using program on OUTFILE (Line 14).

11. The program is taken out of Priority Alert (see Line 15).

12. The address of the IORW on the top of the File List associated with OUTFILE is stored in index register 15 (see Line 16).

13. The IORW on the top of the File List associated with OUTFILE is removed from that list and the next IORW on the list (if any) is moved to the top of the list (see Line 17).

14. The IORW that was at the top of the File List associated with INFILE (if any) is linked to the IORW that is to be placed on the File List associated with INFILE by Step 15 (see Line 18).

15. The IORW that was removed from the File List associated with OUTFILE by Step 13 is placed at the top of the File List associated with INFILE (see Line 19).

16. The address of the File List Origin associated with INFILE is placed in the File List Address Field of the IORW that was removed from the File List associated with OUTFILE by Step 13 (see Line 20).

17. The x-control fields of the IORW's linked to the File List of INFILE are initialized by the System Monitor (see Lines 21 and 22).

18. A branch is executed to Step 1 (see Line 23).

Line (3–5)	Label (6–15)	Operation (16–20)	21 25 30 35 40
01		IOCTL	OPEN, OUTPUT, OUTFILE
02		IOCTL	OPEN, INPUT, INFILE
03		MRZR	*, OUTIORWA+23
04	START	GET	INFILE
05			(Process last record made available)
06		BXPA	*+1
07		ZA	INFILE, X15
08		ZA	0+X15, INFILE
09		ZA	+OUTFILE, 5+X15
10		ZA	OUTFILE, 0+X15
11		ZA	X15, OUTFILE
12		ZA	+OUTFILE, X15
13		B	/MIO/
14		PUT	OUTFILE
15		BXPA	*+1
16		ZA	OUTFILE, X15
17		ZA	0+X15, OUTFILE
18		ZA	INFILE, 0+X15
19		ZA	X15, INFILE
20		ZA	+INFILE, 5+X15
21		ZA	+INFILE, X15
22		B	/MIO/
23		B	START
24			

Figure 117. Two File/Three Area Overlap Programming

Two File/Two Area Overlap Programming

A programming scheme that allows one input file and one output file that share the same two input/output areas to be processed in a manner that provides for the overlapping of read/write and processing operations is discussed in the material that follows. This scheme is efficient if the time required to process a record read by the IOCS is relatively short compared to the time required to read the record (GET) from the input file or write the record (PUT) on the output file. This scheme is only applicable to files made up of unblocked records.

To use this scheme, the program is initialized as follows: (All labels used in this example are for illustrative purposes only.)

1. Two files, one an input file labeled INFILE, the other an output file labeled OUTFILE, are defined.
2. Two input/output areas, one labeled AREA1, the other labeled AREA2, are defined.
3. Two IORW's are generated and labeled; one by means of a DTF IOAREAS entry included in the DTF statement defining INFILE, with AREA1 as its first operand and INIORW as its fourth operand; the other by means of a DTF IOAREAS entry included in the DTF statement defining OUTFILE, with AREA1 as its first operand and OUTIORW as its fourth operand.
4. A one-position field labeled SWITCH is defined.
5. INFILE and OUTFILE are opened (see Lines 1 and 2 of Figure 118).

Once the program has been initialized, the two files are processed as follows:

1. The character A is placed in SWITCH (see Line 3).
2. The low-order address of AREA1 is made the B-address of the input/output instruction in OUTIORWA (see Line 4).
3. The Traffic Bit in OUTIORWA is set OFF (see Line 5).
4. The IOCS is requested to make the next logical record from INFILE available in the input/output area whose address appears as the B-address of the instruction contained in INIORWA (see Line 6).
5. The low-order address of AREA2 is made the B-address of the input/output instruction contained in INIORWA (see Line 7).
6. A branch is executed to PROCESS if the sign of the contents of SWITCH is minus (see Line 8).
7. The contents of SWITCH is subtracted from itself (see Line 9).
8. The IOCS is requested to make the next logical record from INFILE available, on a deferred basis, in the input/output area whose address appears as the B-address of the instruction contained in INIORWA (see Line 10).
9. The IOCS is requested to write on OUTFILE the last record processed by the using program (see Line 11).
10. The low-order address of AREA2 is made the B-address of the input/output instruction contained in OUTIORWA (see Line 12).
11. A branch is executed to PROCESS if the sign of the contents of SWITCH is minus (see Line 13).
12. The IOCS is requested to write on OUTFILE, on a deferred basis, the last record processed by the using program (see Line 14).

Line (3–5)	Label (6–15)	Operation (16–20)	21 25 30 35 40
01	START	IOCTL	OPEN, INPUT, INFILE
02		IOCTL	OPEN, OUTPUT, OUTFILE
03		ZA	+1, SWITCH
04		MLNA	+AREA1, OUTIORWA+14
05		MLZS	@S@, OUTIORWA+23
06		GET	INFILE
07		MLNA	OUTFILE+23 INIORWA+14
08		BZN	PROCESS, SUITCH, B
09		ZS	SWITCH
10		GET	INFILE, DEFER
11		PUT	OUTFILE
12		MLNA	INFILE+23, OUTIORWA+14
13		BZN	PROCESS, SWITCH, B
14		PUT	OUTFILE, DEFER
15		R	LOOP
16	PROCESS	SAR	PROCEXIT+5
17	(Process Here)		
18	PROCEXIT	B	0
19		END	START
20			

Figure 118. Two File/Two Area Overlap Programming

13. A branch is executed to the instruction on Line 6 (see Line 15).

14. The contents of the B-address register are placed in the instruction on Line 18 (see Line 16).

15. The current data record is processed (see Line 17).

16. A branch is executed to the address saved by the instruction on Line 16 (see Line 18).

Appendix B: Modification of GET and PUT Macro-Instructions

The coding generated by all GET and PUT macro-instructions includes the sequence of instructions shown in Figure 119.

Line	Label	Operation	
0.1		B,X,P,A,	/P,E,T,/,
0.2		D,C,W,	F,I,L,E,N,A,M,E,
0.3		D,C,	@∝@
0.4		DC,W,	@,b,@
0.5			

Figure 119. GET/PUT Calling Sequences

The instruction shown on Line 1 of Figure 119 causes the user's program to be removed from Priority Alert and a branch to be executed to the Resident IOCS at /PET/. The label FILENAME shown on Line 2 must be the same label that appears as the operand of the DTF Header Line entry for the file. The character represented by the alpha symbol shown on Line 3 becomes the d-character of the input/output operation requested by the relevant GET or PUT macro-instruction. If this operation is a read operation the DC statement shown on Line 3 is assembled containing the character R. If this operation is a write operation the DC statement is assembled containing the character W. The DCW statement shown on Line 4 indicates whether the relevant GET or PUT macro-instruction is a GET FILE,DEFER or PUT FILE,DEFER. The statement is assembled containing an S if this is the case. It is assembled containing a blank character if this is not the case.

The programmer may alter the d-character of the input/output operation discussed above. He may do this by placing in the operand field of the DC statement shown on Line 3 of Figure 119 one of the following characters:

CHARACTER	RESULTANT INPUT/OUTPUT OPERATION
$	Read to interrecord gap or end of core.
X	Write to end of core.
Q	Alter read instruction to a No Operation (NOP) instruction.
V	Alter write instruction to a No Operation (NOP) instruction.

If the programmer changes the character in the operand field of the DC statement shown in Figure 119 to $ or X, he must also change the channel character in the IORW at the top of the File List associated with the affected file. This channel character is assembled as an @ (i.e., overlap mode, channel 1) or as an (i.e., overlap mode, channel 2). This channel character must be changed to a % (i.e., non-overlap mode, channel 1) or a □ (i.e., non-overlap mode, channel 2). The reason for this change is that Read/Write to End of Core operations must always be executed in non-overlap mode on the IBM 1410. This channel character must be altered before the modified GET or PUT macro-instruction is encountered in the using program.

Appendix C: IOCS Console Messages

This section provides explanations of the IOCS console messages that are of interest to programmers who use the IOCS.

Error Messages

All IOCS error messages conform to the format shown in Figure 120.

The message number shown in Figure 120 is constant. The two-character mnemonic EE indicates the type of error that has occurred (e.g., DC indicates that a Data Check has occurred). The BCD code of the Channel Status Character indicates which channel status indicator(s) have been turned on as a result of the error. If the input/output operation that resulted in the error was a tape read operation, the last field of the error message contains an integer equal to the number of characters transferred into core storage as a result of the operation. If the opera-

Message Number	Error Type	Channel Status Character	Input/Output Instruction	Tape Input Record Length or Disk Address
10101	EE	s	xxxxxxxxxx	----------

Figure 120. IOCS Error Message Format

	Error Type	Channel Status Character	Affected Input/Output Device Type	Explanation	Action
Not Ready	NR	1	Tape, disk, card reader, card punch, printer.	The relevant device (e.g., module of 1301 Disk Storage) is not ready.	Ready the device.
	NR	/	Disk	The IBM 7631 File control is not ready.	Ready the 7631.
	NR	J	Disk	The addressed access mechanism is inoperative.	Ready the access.
Data Check	DC	4	Tape, disk, card reader, card punch, printer	Tape Input Operation: (a) If record length is greater than 12 characters a backspace followed by a read operation has been executed 99 times. (b) If record length is 12 characters or less, ten consecutive noise records have been read without changing the B-address of the affected input instruction. This indicates that the relevant tape unit has been set to the wrong density. Tape Output Operation: A backspace, a skip and blank tape, and a write operation have each been executed, in sequence, 25 times. Disk: The relevant input/output operation has been re-executed 4 times. Card Reader: The last card read is in error. Card Punch: Machine parity error. The card is not punched. Printer: Machine parity error. The line is not printed.	(a) None. (b) Set the relevant tape unit to the correct density. None. None. Run out card deck and reload. Make the card reader Ready. Make the device Not Ready, then Ready. Make device Not Ready, then Ready.
	DC	5	Disk	The relevant input/output operation has been re-executed 5 times.	None.
	DC	M	Tape, disk, card reader	See "DC4" above.	See "DC4."
	DC	@	Tape	See "DC4" above.	See "DC4."
Condition	UC	8	Card punch, printer	Card Punch: An incorrect card has been punched. The erroneous card has been selected into stacker 0. Printer: A timing error or hammer fire check has occurred. The line is not printed.	Make the device Not Ready, then Ready. Make the device Not Ready, then Ready.
Wrong Length Record	WL	-	Tape, disk card reader, card punch, printer	Tape Input Operation: The input operation has been re-executed 9 times. Tape Output Operation: The output operation has been re-executed 25 times. Disk: The input/output operation has been re-executed four times. Card Reader: The input/output operation has not been re-executed. Card Punch: The input/output operation has not been re-executed. Printer: The input/output operation has not been re-executed.	None. If the error persists on the same record, the program should be terminated. None. None. None. None.
	WL	Q	Tape	Tape Input Operation: A tape mark has been read by the IOCS, but it has been determined that the end of the relevant data file has not yet been reached.	None.
No Record Found	NF	Z	Disk	The input/output operation has been re-executed 4 times, the access mechanism has been recalibrated, and the operation has been re-executed an additional four times.	If the message is repeated for the same record, the program should be terminated.
No Track Found	IT	V	Disk	The relevant access mechanism has been recalibrated and the input/output operation has been re-executed one time.	If the message is repeated for the same record, the program should be terminated.
Mode Check	MD	V	Disk	Data check, no transfer, and condition indicators have resulted from a disk operation.	None.
Circuit Check	CC	8	Disk	A 7631 circuit check has occurred. This input/output operation has been re-executed one time.	None.
	CC	Q	Disk	See "CC8," above.	See "CC8."
1301 Circuit Check Invalid Operation Write Disk Check Without Mode	OP	9	Disk	The input/output operation has been re-executed one time.	If the message is repeated for the same record, the program should be terminated.
UNCTL FILE, SSF,d	SS	b	Card Reader	Two UNCTL FILE,SSF,d macro-instructions have been executed without an intervening Read A Card instruction that contains 9 in the units position of its x-control field. Two Read a Card instructions with 9 in the units position of their x-control fields have been executed without an intervening UNCTL FILE,SSF,d Macro-instruction.	None.
Write Inhibit	WI	b	Disk	Write operation was attempted, but write inhibit switch on 7631 is ON.	Turn Switch OFF, or terminate program.
Unknown Error	UEP		Disk	The IOCS cannot determine the exact nature of the error. It is looping on the operation that caused it.	Terminate program.

Figure 121. Error Type — Channel Status Character

tion was a disk operation, the last field contains the relevant disk address, in the form shown in Figure 111.

The various combinations of characters that may appear in the error type and channel status character fields of IOCS error messages are described in Figure 121.

Tape Label Messages

The IOCS provides three tape label messages. These messages are described in Figure 122.

Message	Explanation	Operator Action
10102 cu xxxxx	The block count of the affected input tape trailer label does not equal the block count accumulated for the reel by the IOCS. (cu is the channel and unit of the affected tape; xxxxx is the difference between the IOCS block count and the block count, on the reel.)	None
30101 cu	One or more fields of the affected input tape header label do not match the corresponding field(s) of the File Table Extension used to check the label. (cu is the channel and unit of the affected tape.)	The IOCS can be directed to accept the tape, or to re-execute the label check with a different reel mounted. 1. Press Inquiry Request key. 2. To re-execute, enter $8R; to accept, enter $8A. 3. Press Inquiry Release key.
30102 cu	The retention period of the relevant output tape header label indicates that the tape should not be written on. (cu is the channel and unit of the relevant tape.)	The IOCS can be directed to accept the tape, or to retry the label check with a different tape mounted. 1. Press Inquiry Request key. 2. To re-execute, enter $8R; to accept, enter $8A. 3. Press Inquiry Release key.

Figure 122. IOCS Tape Label Messages

Appendix D: Work Files

A work file is defined as a file that is used both as an input file and an output file. The capability to handle such a file is a feature of the IOCS. Only one DTF statement is required to define a work file, because the File Table generated by a DTF statement does not indicate whether the file is an input or an output file. However, the programmer must be careful to include all the DTF entries in the work file DTF statement that are appropriate to both an input and an output file of the relevant type (e.g., tape or disk).

The IOCS determines whether a work file is to be considered an input or an output file, when the file is opened by means of either the IOCTL OPEN,INPUT or the IOCTL OPEN,OUTPUT macro-instruction. For this reason, when the file is to be switched from an input file to an output file, or from an output to an input, the file must be closed by means of the appropriate IOCTL CLOSE macro-instruction and then reopened. The IOCTL OPEN macro-instruction used to reopen the file indicates the file's new status.

Appendix E: Modification of the File Table Extension

The contents of the fields of the File Table Extension can be altered if the programmer wants: (a) an input header label to be checked against information other than that currently contained in the relevant File Table Extension (i.e., the information originally supplied by the DTF LABEL entry and updated where appropriate by the IOCS), or (b) an output header label to be written from information other than that currently contained in the relevant File Table Extension.

The following procedure can be used to alter the information contained in the fields of the File Table Extension:

1. Before the affected file is opened, the user's program reads a card containing the desired information from the system's Standard Input Unit. (See the *System Monitor* publication.) The format of this card is left to the user's discretion.

2. The desired information is then moved from the SIU input area (i.e., the area whose high-order position is labeled /CRD/) into the relevant field or fields of the File Table Extension.

3. The affected file is opened by means of an IOCTL OPEN macro-instruction. The label is then checked or written by the IOCS, using the new information contained in the File Table Extension.

Index

Where more than one page reference is given, major reference appears first.

Reader's Comments

IBM 1410/7010 Operating System (1410-PR-155)
Basic Input/Output Control System--1410-IO-966

Form C28-0322-1

From

Name _____

Address _____

Your comments regarding the completeness, clarity, and accuracy of this publication will help us improve future editions. Please check the appropriate items below, add your comments, and mail.

	YES	NO
Does this publication meet the needs of you and your staff?	____	____
Is this publication clearly written?	____	____
Is the material properly arranged?	____	____

If the answer to any of these questions is "NO," be sure to elaborate.

How can we improve this publication? Please answer below.

☐ Suggested Addition (Page , Timing Chart, Drawing, Procedure, etc.)

☐ Suggested Deletion (Page)

☐ Error (Page)

COMMENTS:

No Postage Necessary if Mailed in U.S.A.

C28-0322-1

STAPLE

FOLD

FOLD

FIRST CLASS
PERMIT NO. 81

POUGHKEEPSIE, N. Y.

BUSINESS REPLY MAIL
NO POSTAGE STAMP NECESSARY IF MAILED IN U. S. A.

POSTAGE WILL BE PAID BY
IBM CORPORATION
P.O. BOX 390
POUGHKEEPSIE, N.Y.

ATTN : PROGRAMMING SYSTEMS PUBLICATIONS
DEPARTMENT D91

FOLD

FOLD

CUT ALONG LINE

Printed in U.S.A. C28-0322-1

IBM

International Business Machines Corporation
Data Processing Division
112 East Post Road, White Plains, New York

STAPLE